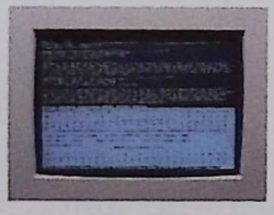
$5.00 March/April 1990

THE PICK® USER DIGEST

An educational journal dedicated to users of PICK and PICK/UNIX computer systems.

Esprit 400. New Technology in ASCII/ANSI Displays.

Dual-host, simultaneous updating applications. Window: DEC All-In-One. Full screen: SCO MultiView.



Benefit

See two complete applications.

Run two 24-line terminal screens including a status line and a message line, or operate a 52-line display.

Benefit

View 132- and 80-column applications together, the way they were meant to be viewed.

See 132-character programs in 132-character format and 80-character programs in 80-character format — a technology first.

Benefit

User-friendly windowing of all standard applications. Greater information access. More productivity.

Use WindowFrame to tailor your window screen. Pan, Zoom, Blank and Move your window with on-screen menus and simple keystrokes.

Benefit

Work with high quality graphics.

The Esprit 400G model features Tektronix 4010/4014, Hercules graphics and 8 pages of memory.

Esprit and Esprit 400 are registered trademarks of Esprit Systems, Inc. All other product names are trademarks of their respective holders.

For too long ASCII/ANSI technology has been limited to incremental improvement. Refresh rates a few hertz higher. A couple more emulations. Slightly larger character cells.

That was then.

Introducing the Esprit 400. An ASCII/ANSI/PC terminal featuring major technology breakthroughs. With benefits that no other terminal offers.

Like easy-to-use windowing with total display presentation flexibility. Two complete 26-line application screens. A 52-line screen. Ability to mix 132- and 80-character formats on the same line. And dual-host simultaneous updating.

In short, one Esprit 400 equals two independently configurable and separately operating terminals — on a single display.

Esprit 400 also combines international design standards with leading ergonomic features like 78Hz refresh, overscan and a choice of keyboards. A full complement of emulations including DEC VT320, WY-60 and PC-Term lets Esprit 400 run existing applications under Unix, Xenix, VMS, Pick, DOS and others.

Now for the same price as yesterday's terminal, get the new technology of Esprit 400.

For your free demonstration please call 800·645·4508 (516·293·5600 NY State).

Esprit

INSIDE

Cover design by Patricia Goss-Millington

Publisher's Note:

The **PICK USER DIGEST** is directed to the users of PICK® and PICK/UNIX based computing systems. The scope of the magazine is educational with presentations offered by technical experts in the PICK and PICK/UNIX field.

The content is intended to assist users of these systems by expanding on technical aspects of the system as well as providing software profiles where available. In addition, the readers may take advantage of the program utilities offered in the form of source code. Although we make every effort to verify the contents of this publication, we cannot accept the responsibility or liability for error of technical content.

The **PICK USER DIGEST** is published bi-monthly by Thurman Marketing Services, Inc., publishers of the NEWS & REVIEW newspaper to the PICK an PICK/UNIX environment.

Subscription rates: (Annual) U.S., Canada and Mexico - $40 US. Other countries - $60 US.

For advertising information call: (714) 855-4442. Fax: (714)380-3942. We have the lowest rates available.

Articles may be submitted to the editor for consideration. Floppy diskettes in either PICK or Macintosh Microsoft WORD format is preferable.

Letters to the Editor

Send letters to: Editor • Pick User Digest • 23181 Verdugo #104A • Laguna Hills, Ca. 92653

Pick Relational?

Dear Pick User Digest:

I found the recent article "PICK" is not a Relational Data Base Management System" [Jan/Feb Issue] quite disturbing. Essentially, the author argued that Pick's failure to be a Relational Database Management System (RDBMS) is somehow advantageous. It is clear that the intended advantages of the relational model were not properly investigated. In pursuance of fair-play, it is only appropriate that we examine Codd's objectives when he first proposed the relational model (1):

Data Independence Objective: The relational model was developed to "yield maximal independence between programs on the one hand and machine representation and organization of data on the other". Codd was calling for a separation between the physical representation of data and the logical view of that data. The separation of these layers is termed data independence; when an application views data only through its logical layer, that application will run independently of any changes to the underlying physical representation. The relational model, then, provides a logical-level view of data to the application.

PICK achieves a degree of data independence in INFO/ACCESS; data is viewed through dictionaries which provide a logical "naming" scheme for data attributes. The notion of a logical layer, however, is lost when programming in Data/Basic. The Data/Basic programmer views data in terms of its physical (rather than logical) representation. For example, a statement of the form:

READV Name FROM Staff, Staff#,1 THEN

makes explicit reference to the "ordinal position" of data attributes. If the physical structure of a file subsequently changes, thus affecting the ordinal position of attributes, each program referencing that file may also need to be modified and re-compiled.

Worse still, the Data/Basic programmer is able to exploit knowledge of the physical storage of data and produce terrible data dependent code of the form:

WRITEV Name:CHAR(254):Age:CHAR (254) :Job Title ON Staff, Staff#

Thus, a significant disadvantage of PICK over the relational model is that it permits (and often even requires) the programmer to exploit knowledge of the physical representation of data.

Communicability Objective: Codd wanted the relational model to be based upon a small number of concepts, and for the representation and manipulation of data to be simple and intuitive. This enabled people to learn the model easily, and to describe databases based upon the relational model in simple terms. This quest for simplicity explains Codd's "first normal form assumption" that all attributes be atomic (i.e. single valued). PICK has clearly demonstrated the utility of multi-valued data. Codd, however, argues that such extentions to the basic relational model entail a great deal of theoretical work to determine the corresponding extentions to the set of relational operators. Several researchers have examined extension of the relational model in this way and have argued quite convincingly that it is indeed possible to "relax the first normal form assumption" whilst remaining faithful to the relational model (3). PICK, however, has not benefited from such research and hence does not provide the necessary set of data operators. For example, sub-values do not have a corresponding READSV statement in Data/Basic.

Set-Processing Objective: The relational model supports "set processing" (2); operators are provided for processing multiple records at a time, hence the programmer is not forced to code in terms of iterative loops. The relational model is based upon a well-understood set of operators which operate upon sets of data rather than record-at-a-time. Thus, the relational operators exhibit more expressive power (i.e. they do more) than the READ and WRITE statements of Data/Basic. Indeed, the relational calculus (a notation proposed by Codd for manipulating relations), provides a higher level (declarative) programming language than is available in PICK, affording three advantages: (1) Programs will be more compact and easier to write, (3) Relational calculus is optimizable implying greater performance.

Conclusion: Experience has shown that PICK provides facilities that afford the application developer a high degree of flexibility and productivity. It is equally evident, however, that PICK suffers from a number of significant weaknesses that must be addressed in order to satisfy the application developments needs of the future. It is precisely these weaknesses that we at ADDS are now addressing.

References:

(1) Codd, E.F. A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, 13,6, June 1970, p377-387.

(2) Codd, E.F. Relational Database: A Practical Foundation for Productivity, The 1981 ACM Turing Award Lecture, Los Angeles, Nov. 1981.

(3) Fischer, P., and Thomas, S. Operators for Non-First-Normal-Form Relations. In Proceedings of the seventh International Software Applications Conference, Chicago, Nov. 1983, p186-196.

(4) Marz, G. PICK is not a Relational Data Base Management System, PICK USER DIGEST, Jan/Feb 1990, p47-49.

- Anthony P.J. Lauder
Applied Digital Data Systems (ADDS)

Ed: Keep us posted, perhaps our readers would like to hear more about these enhancements from your company.

(Continued on page 4)

The Pick® Master Distributor You Can Trust

Pick Systems now allows you the advantages of an authorized Pick Dealer while developing a direct relationship with the world's leading manufacturer of Pick Performance Technology.

Monolith's Pick Master Partner Program gives you everything you need to be successful.

Continuation of Dealer Advantages

As a master distributor, Monolith offers superior discounts while your authorized dealer status and benefits remain fully intact.

Factory Direct

Factory Direct sales, shipment, and support means no inefficient and costly middle man between you and the solutions you need. Monolith designs and builds its solutions specifically for Pick. As a result, Nobody ships more products to the Pick Industry, nobody.

A Pick Distribution Revolution

Same Day Shipment of Operating Systems...Guaranteed. Call 800/ALL-PICK.

Dealer Dedication

Monolith's reputation begins with its proven track record of supporting a worldwide dealer network spanning 30 countries. Only an actual history of dedicated dealer-only sales and support should earn your trust.

Marketing Support

From a generous demonstration equipment program to comprehensive nationwide hardware and software leasing, the Pick Master Partner Program has it all.

Technical Support

As a manufacturer and developer totally dedicated to Pick, no one can support you better.

International On-site Maintenance

As a co-manufacturer with NCR, Monolith taps a field service network of 16,000 engineers to provide one of the most extensive Pick specific support and maintenance programs in the industry.

Solutions

Peripherals - Monolith's revolutionary I/O, caching, and tape subsystems provide new capabilities, increased performance, and greater flexibility to unleash the power of your Pick micro.

Systems - Our advanced performance systems run circles around common PC's. The NCR based 8000*i* series provides seamless expandability from 3 to 33 users fueled by Monolith's intelligent I/O and caching technologies. The "Monolith, Built by NCR" label means superior quality, onsite warranty, and total customer confidence.

Software - Monolith publishes software too! Software in tune with todays market. Take for example, Desqtop, the standard in Office Automation software provides powerful multiuser communications, time management and organizational tools. And FlashCache, a revolutionary software solution that utilizes extended memory to optimize system performance.

It's Time to Get Serious

☐ YES! Send my copy of the Monolith Pick Master Partner Program Manual at no obligation to me.

Name _____

Company _____

Address _____

City/State/Zip _____

Phone ( ) _____

For faster service call 800/ALL-PICK

Monolith CORPORATION

World Headquarters 335 South White Street Wake Forest, NC 27587
800/ALL-PICK 919/556-6664 FAX: 919/556-1920

Dear PICK USER DIGEST:
In response to the letter from Roy Harwell of MONOLITH CORPORATION (Jan/Feb issue) regardinging Mr. Alexander's article about nested loops. The question discussed was whether the order of loops can ever make a difference in speed when one is to be nested within the other and one of them is larger. At the risk of seeming to speak for Mr. Alexander, I offer the following comment:

When first described, it seemed that there should be no difference at all, as well argued by Mr. Harwell. But when he suggested that there may be some anomaly in PICK that could change the actual results - I saw the light. There IS a somewhat famous practical issue in PICK performance: an anomaly which wouldn't be likely to show up in the test program Harwell ran, but could in a larger, real-world piece of code:

FRAME FAULTING.

And this thought makes me understand the wisdom (as a rule-of-thumb) of the Alexander advice to place larger loops inside . . .

If the section of object code that covers all of both loops crosses frame boundaries, then there will be a time penalty for doing this crossing, and the penalty will be related to the number of times the crossing occurs. Where these borders occur can be seen by compiling with an M option ('M' for map), or you can examine the actual object code where and how it lies.

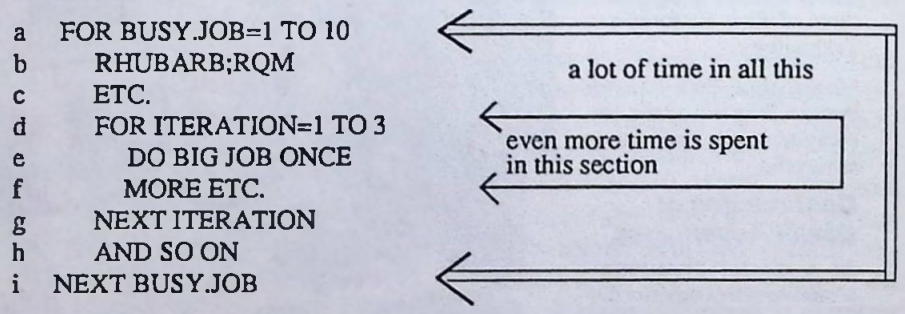
Consider this example . . . The outer loop is for three iterations, and the inner is for ten. An imaginary listing for this code is shown as follows:

```
a    FOR ITERATION=1 TO 3
b       RHUBARB; RQM
c       ETC.
d       FOR BUSY.JOB=1 TO 10
e          DO BIG JOB ONCE
f          MORE, ETC.
g       NEXT BUSY.JOB
h       AND SO ON
i    NEXT ITERATION
```

(Note that most time is spent in code execution of lined d thru f).

If this is all in one frame, then Harwell is always correct: there can be no difference. If there ARE breaks within, consider the possibilities:

one break in zone:	number of frame faults:
a	3
b	3
c	3
d	30
e	30
f	30
g	30
h	3
i	3



one break in zone:	number of frame faults:
a	10
b	10
c	10
d	30
e	30
f	30
g	30
h	10
i	10

And of course breaks don't just occur at source code line borders. Many source-code-simple statements actually become complex at the object code level; so breaks can occur in the midst of these lower-level bytes. And also note that the smaller the busy part is, the smaller the chance that it contains one or more breaks. The best situation is that it is all in one frame so that the busiest part is NOT the one with the penalty border(s) inside. If only the outer loop hits a break, the penalty occurs a smaller number of times.

Thus, the Alexander advice to push busier loops deeper will work to increase the chances of better performance. It is not a make-or-break rule, but it IS the slightly better of what first appeared to be identical alternatives. Thanks to both Steve Alexander and Roy Harwell for making me think these thoughts (a timeslice saved is a . . .). And thanks to those who have finished reading this letter.

- Jim Carroll, Consultant
Jenkintown, Pa.

Dear PICK USER DIGEST:
Regarding the letter from Roy Harwell relating to the use of indexes in FOR . . . NEXT loops. I think this point needs some clarification. The point I would like to make is that it really does matter how you nest these loops. The program below demonstrates this:

```
FOR OUTER = 1 TO 1000
   FOR INNER = 1 TO 10
   NEXT INNER
NEXT OUTER
```

Compiled as: BASIC BP TEST (A)
gives:

```
001  03          LOADA      OUTER  }
001  07          LOAD 1            }
001  07          LOAD 1            } Initialise outer loop
001  2d          SUBTRACT          }
001  5f          STORE             }
001       *1009
001  05          LOADN      1000                          ] Start of outer loop
001  03          LOADA      OUTER                         ]  executed 1000 times
001  07          LOAD 1                                   ]
001  1b          FORNEXT    *2009                         ]
001  01          EOL                                      ]
002  03          LOADA      INNER  }                      ]
002  07          LOAD 1            }                      ]
002  07          LOAD 1            } Initialise inner loop ]
002  2d          SUBTRACT          }                      ]
002  5f          STORE             }                      ]
002       *1010                          }                ]
002  05          LOADN      10          }                 ]
002  03          LOADA      INNER       }                 ]
002  07          LOAD 1                 } Inner loop, executed ]
002  1b          FORNEXT    *2010       } 10 times        ]
002  01          EOL                    }                 ]
003  06          BRANCH     *1010       }                 ]
003       *2010                          }                ]
003  01          EOL                                      ]
004  06          BRANCH     *1009                         ] End of outer loop
004       *2009
```

We can see from the above that the total number of instructions is made up of:

1 initialization for the outer loop
1000 FORNEXT loops for the outer loop
1 initialisation for the inner loop
10 FORNEXT instructions for the inner loop

Suppose we let the number of outer loops = N, and the number of inner loops = "M", then the total number of instructions is of the order 1 + N + NM. I.e. we end up with 1 inner loop, and a total of N*M FORNEXT loops. It's fairly obvious that the total number of instructions becomes a function of the number of times we must initialise the inner loop, so in order to achieve the best time (all other things being equal), we want the SMALLER index on the outer loop, and the LARGER index on the inner loop.

For the example, if N = 1000, and M =

(Continued on page 6)

10 then we have of the order:

1 + 1000 + 1000*10 = 10,011

instructions, whereas if we swap the indexes around (the generated code will be identical except for the swapping of indexes as noted by Roy Harwell) the number of loops is now:

1 + 10 + 1000*10 = 10,011

which is less, so it really DOES matter which index is used.

This is not an anomaly of PICK Basic and holds true for almost all compilers, be it whatever language (the exceptions may be where the compiler is an optimising compiler and can somehow fold the loops).

If you really want to convince yourself the above is true, run the following program:

```
COUNTER = 1
FOR I = 1 TO 1000
   COUNTER = COUNTER + 1
   FOR J = 1 TO 10
      COUNTER = COUNTER + 1
   NEXT J
NEXT I
PRINT COUNTER
```

Then swap the loops and run it again.

- Bryan Buchanan, Parthenogenesis, Pty Ltd - Lawnton, QLD Australia

Dear PICK USER DIGEST:

I enjoyed the letter from Roy Harwell, about nested loops. Several people called me to report similar observations (they thought) on their machines.

Here is a printout of another program called FAST.SLOW and the 'pseudo assembler' code it generates. This was done on an Everex 386 machine running PICK 3.0. As you can see, the 'pseudo assembler' is not exactly the same for the two loops. Furthermore, the times for the two loops are considerably different on my machine. The slow loop takes about 13 seconds, while the fast one runs in about 7 seconds on the Everex 386.

```
FAST.SLOW
START = SYSTEM(12)
FOR I = 1 TO 10000
   FOR J = 1 TO 3
      X = 3
   NEXT J
NEXT I
CRT 'SLOW ':SYSTEM(12) -START
*
START = SYSTEM(12)
FOR J = 1 TO 3
   FOR I = 1 TO 10000
      X = 3
   NEXT I
NEXT J
CRT 'Fast ':SYSTEM(12) - START
```

Pseudo Assembler: (FAST.SLOW)

```
FAST
001 05      LOADN       12
001 88      SYSFLAG
001 18      STOREA      START
001 01      EOL
002 03      LOADA       I
002 07      LOAD 1
002 07      LOAD 1
002 2D      SUBTRACT
002 5F      STORE
002
002 05      LOADN       10000
002 03      LOADA       I
002 07      LOAD 1
002 1B      FORNEXT     *2009
002 01      EOL
003 03      LOADA       J
003 07      LOAD 1
003 07      LOAD 1
003 2D      SUBTRACT
003 5F      STORE
003
003 05      LOADN       3
003 03      LOADA       J
003 07      LOAD 1
003 1B      FORNEXT     *2010
003 01      EOL
004 05      LOADN       3
004 18      STOREA      X
004 01      EOL
005 06      BRANCH      *1010
005     *2010
005 01      EOL
006 06      BRANCH      *1009
006     *2009
006 01      EOL
007 05      LOADN       12
007 88      SYSFLAG
007 16      LOAD        START
007 2D      SUBTRACT
007 09      LOADS       "Slow"
007 10      MULTICAT
007 5D      CRTCRLF
007 01      EOL
008 01      EOL
SLOW
009 05      LOADN       12
009 88      SYSFLAG
009 18      STOREA      START
009 01      EOL
010 03      LOADA       J
010 07      LOAD 1
010 07      LOAD 1
010 2D      SUBTRACT
010 5F      STORE
010     *1011
010 05      LOADN       3
010 03      LOADA       J
010 07      LOAD 1
010 1B      FORNEXT     *2011
010 01      EOL
011 03      LOADA       I
011 07      LOAD 1
011 07      LOAD 1
011 2D      SUBTRACT
011 5F      STORE
011     *1012
011 05      LOADN       10000
011 03      LOADA       I
011 07      LOAD 1
011 1B      FORNEXT     *2012
011 01      EOL
012 05      LOADN       3
012 18      STOREA      X
012 01      EOL
013 06      BRANCH      *1012
013     *2012
013 01      EOL
014 06      BRANCH      *1011
014     *2011
014 01      EOL
015 05      LOADN       12
015 88      SYSFLAG
015 16      LOAD        START
    2D      SUBTRACT
    09      LOADS       "Fast"
    10]     MULTICAT
    5D      CRTCRLF
    01      EOL
    01      EOL
    45      EXIT
```

- Steve Alexander, Triadyne of America, San Diego, Ca.

Dear PICK USER DIGEST

Hats off to Brian Stone for his informative article concerning the now-apparent benefits of EMS 3.0! My apologies to him for my "indicting" him as he clearly knows his stuff.

Indeed, the software vendor, referred to in my previous letters, installed 3.0 EMS on her customer's machine which, according to the customer: "sped things up a lot". Also, the software vendor rewrote her "number-crunching" forms interpreter (a clever but slow program), transforming it into an efficient code-generator. And, the sofware vendor is upgrading her screen interpreter to use less workspace and run faster. A lot of work, but beneficial to all of her customers in the long run.

It was my fault for assuming that the technical guru's knowledge about 3.0 EMS was complete and accurate. The guru did tell me, however, that he was "pretty sure" that 3.0 EMS handled READS only, and not WRITES. My only goal in all this was not to indict Mr. Stone, but rather to get at the truth in the matter, which we now have thanks to Mr. Stone.

I do have another question for Mr. Stone, however, concerning the overall power of PC's. Since I am a software person, and deliberately know very little about hardware, I have no idea whether another rumor being spread around is true, that being: "The bus structure on PC's is incapable of handling large numbers of interactive users effectively. The bus structure is slow and primitive compared to those of many mini-computers". Is this true? If we put 33 active users on a best-configured 386, will the bus itself become the bottleneck?

Finally, my apologies to Dave Zigray of Pick Systems for not trusting his response to my first letter. After comprehending Mr. Stone's article I now see that Mr. Zigray was fully accurate in his response. As a result of all this, PC's running PICK's 3.0 have been re-placed on my list of systems worth selling. . . .

- Dave Ramette, ACE Software
Laguna Beach, Ca.

- END -

THE BENEFITS ARE IN YOUR HANDS

VIA Duct easily connects PICK to DOS, DOS to PICK, and PICK to PICK. The boundaries disappear with an intuitive user interface. VIA Duct allows you to Transfer, Capture, Convert, Emulate, Print, and Hot Key between environments seamlessly and effectively. The power of Virtual disks, Virtual Windowing, and automatic conversion of PICK data into LOTUS 1-2-3 and other DOS formats are within your grasp.

VIA Duct is the Answer

VIA Duct

VIA Systems
Formerly INTERACTIVE SYSTEMS

655 Southpointe Court
Suite 100
Colorado Springs, CO 80906
(719) 579-6800
FAX (719) 576-7246

Why PCs Do Not Run PICK As Well As Minis

by Ian Sandler - Director Software Engineering, General Automation
and Author of "THE PICK PERSPECTIVE"

Reading some of the articles printed in this, and other similar magazines, I am left with the distinct impression that you can buy a PC for peanuts, load it up with inexpensive hardware and a copy of PICK, and outperform the low end PICK Minicomputers sold by companies such as ADDS, Alpha Micro, Altos, General Automation, etc. There are even benchmarks quoted which claim that some PCs can outperform fairly large and expensive Minis.

Let us look at these claims for a moment. Most PICK PCs run with 80286 or 80386 CPUs, which perform at roughly the same speed as the processors used in equivalent Minis. Therefore it is not really surprizing that a PICK PC with a 25mhz 80386 CPU will not run single user, CPU intensive tasks about as fast as a Mini with a 20mhz 60820 CPU.

One can buy intelligent disk I/O subsystems for a PICK PC which will reduce the time taken to read or write a frame of data from around 20 milliseconds (assuming you are using good ESDI or SCSI disks), to one or two milliseconds in many cases. We all know that PICK systems are typically disk bound, so where is the catch? Why won't a PICK PC run 16 or more terminals as fast as a mini?

Unfortunately, there are several performance limitations which the PC salesmen accidentally forget to tell you about. The first of these occurs in the area of terminal I/O. Most add-on serial I/O boards sold for PCs do not work even remotely as efficiently as equivalent boards for Minis. This does not matter much when you have a fast CPU and only a few serial I/O ports. However, once you get beyond ten or so ports, benchmarks start running noticeably slower, even if no-one is using these additional ports.

The mere presence of these serial I/O boards may well slow a system's ability to perform useful PICK computations by a factor of two by the time you get to thirty or so ports. Of course, you can always buy more expensive serial I/O boards, and avoid this problem. Also, if you are only running 16 or so terminals, an 80386 based PC still should have sufficient CPU horsepower, though at a higher cost than you were led to expect.

The next problem occurs with disk and tape I/O. PCs are essentially single user computers, even when they are running PICK. When any user reads a frame from a disk, or writes a block to tape, the entire computer stops until that task is completed. Contrast this with a Mini, which suspends just the task responsible, and allows other users who are not waiting for this specific resource to continue working.

To demonstrate this particular bottleneck, try running the same CPU intensive benchmark on both a PC and a Mini, with and without another process selecting a large file at the same time. When you add the select process, you will see the CPU intensive benchmark running on the PC slow down much more dramatically than the same process running on the Mini.

Even without an active select process running, an

It's A Shame There Aren't As Many Good Instructors As There Are Instructors.

Selecting the right source for Pick® training can be an education in itself. Naturally, you want an instructor who knows this uniquely powerful system inside and out. And you want someone who will teach to your level. What you *don't* want is a semi-retired computer type with surface knowledge. Or a self-styled "expert" that talks down to you. Or, worse yet, some propeller head who spouts only technobabble.

As you can appreciate, good instructors are hard to find. We know, because JES & Associates has taken great pains to assemble the biggest and the best teaching staff in all the Pick® world. Actually, we've had a lot of time to sign up the finest instructors since JES has been doing quality Pick® training longer than anyone else in the business. Every member of our staff is an acknowledged industry expert. Many are respected and well-read authors. And all are excellent teachers.

Why waste your time searching for knowledge when you can get a real education at the Ivy League of Computer Schools! That's JES, specialists in Pick® training and your best choice for UNIX™ classes as well. You'll see proof when you request our comprehensive course catalog, complete with profiles on our twelve top instructors. Call or write today for schedules on upcoming classes in your area and facts on the finest computer training available anywhere.

JES & Associates, Inc.
The Pick® & UNIX™ Training Source
P.O. Box 19274 □ Irvine, CA 92713
(714) 553-8200 □ FAX (714) 553-9779

Pick® is a registered trademark of Pick Systems. UNIX™ is a registered trademark of AT&T Bell Laboratories.

average active PICK system typically generates one or two disk I/Os per active process per second. Therefore, assuming that the PICK PC has sufficient memory to avoid thrashing, with 8 active processes you will lose about 25% of your system's CPU throughput, compared to an equivalent Mini. With 16 active processes a PC will lose about 50% of the system's throughput waiting for other processes to complete disk I/O.

Adding an intelligent disk controller to the PC helps disk reads only if the data being selected is resident in the controller's memory. Otherwise reading data from disk takes slightly longer because of the overhead of searching the controller's cache. Writing data to disk is much faster, but because a typical PICK system reads about four times as much data from disk as it writes to disk, the gains are not as good as you might have been led to expect.

Perhaps the most serious limitation of PICK on a PC is its inability to use more than 640K of memory, except as a buffer between the PICK environment and the disks. The PICK monitor and ABS frames take up roughly 128K. A running PICK BASIC program typically requires somewhere between 8 and 32 frames to avoid thrashing, and twice that number of frames to run efficiently. Your system also requires some space for data frames. These limitations force PC PICK to continue to use 512 byte data frames, rather than the more efficient 1024 byte frames of Advanced PICK, or the 2048 byte frames of most PICK Minis.

ABS frames on a PICK PC are 2048 bytes, just like on a Mini. This means that the PC release has to divide its limited memory into two separate pools, each of which is vulnerable to thrashing. Incidentally, the existence of two separate pools is the reason why PICK PCs run so slowly when running products such as JET or Compusheet at the same time as other users are running ACCESS or BASIC. There are not enough buffers in the ABS pool to avoid thrashing.

The current PICK PC software implementation also has a severe performance bottleneck built into it. It still uses group locks instead of item locks in BASIC. If you have a file with a small modulo, and you use READU or MATREADU statements in your application, you will find that your application wastes a lot of time waiting to be able to access or update items which arein locked groups. The items required are locked out because they accidentally happen to be in the same group as a different item which someone else has locked.

All the Mini-computer implementations use item locks, as does Advanced Pick. Meanwhile, until Pick Systems chooses to solve this problem, if you wish to use PICK on a PC, be sure to size files which are referenced by BASIC READU or MATREADU statements so that they do not contain many items per group. This will minimize the problem, though at the expense of additional disk storage and slightly slower overall system performance.

Do not think I am against the use of PCs for running PICK. I am not. They are excellent for software development. They provide sparkling single user performance, and software written for them is easy to transport to other, larger PICK Minis. They make ideal low end PICK computers.

PICK PCs can provide excellent price/performance for applications with small numbers of active users and large data files. Hard disks for PCs typically cost less than equivalent disks for Minis. However, PICK PCs do not have the same sophisticated backup and restore utilities as the PICK Minis. Therefore PCs are an excellent choice when the bulk of the data is static, and a bad choice when you have to back up or restore more than a few megabytes of data per day.

PICK PCs are also cost effective compared to PICK Minis when the application is unusually CPU intensive. Though here they tend to compare unfavorably with Revelation. (The single-user PICK-like implementation which allows multiple PCs to be networked together).

I do not see much use for expensive intelligent disk controllers for PICK PCs. If you have large data files, and only a few users, they are considerably less effective than software products such as Modular Software's SmartCache product. When running JET or Compusheet at the same time as BASIC or ACCESS, the EMS feature built into the 3.0 PICK PC release is as effective as an intelligent disk controller, and costs no more than a couple of hundred dollars for additional memory to make it effective.

If you are looking for a computer to run a typical sixteen to thirty two user PICK application which will have ten or more active users most of the time it is running, do not buy a PC without careful research. Run benchmarks only on PCs configured exactly as you will buy them. Unused serial I/O ports, (or different serial I/O ports) can make very significant differences to system throughout. Make sure your benchmarks reflect your application, and are multi-user, with as many simultaneous processes as your business problem requires.

You will almost certainly discover that by the time you have finished buying suitable serial I/O boards and a fast enough CPU, you will have to pay more for a PICK PC than you would for a comparable PICK Mini with much better growth potential.

- END -

- ACCESS SHELL -

A Paging Utility for ACESS users

by Dave Ramette, Ace Software

There seems to be a lot of hype in the "industry" lately concerning the entity known as "SQL". It appears that there are some very pronounced shortcomings in SQL. First and foremost, SQL apparently can only represent and refer to "two-dimensional data-tables"-- in other words the dynamic flexibility of "multi-values" and "sub-values" that we take for granted, is void in SQL because they don't meet the "requirements" imposed by one "relational data-base purist" known as Codd. Exploding sort-selects, a most useful feature in ACCESS, would be rejected by Codd, even if (s)he were to wisen up and take a look at what other people in this four-dimensional world are doing.

ACCESS was deliberately designed to be an "inquiry-only" NON-PROCEDURAL language (with the exception of REFORMAT, DELETE coupled with select lists, and EDITOR prestore commands), meaning that no matter how poorly or wrongly an end-user constructs an ACCESS LIST, SORT, SELECT, SSELECT, SUM, COUNT or STAT statement, (s)he can be reassured that no data will be damaged or wiped out.

ACCESS is far easier for an end-user to comprehend, than the PROCEDURAL language SQL. Procedural languages are for programmers, and best of luck to anybody who attempts to explain "nested parenthetical procedural references" to a novice (as most end-users are).

Another major problem with SQL is its apparent ability to gobble up tremendous amounts of disk space. If virtually every major type of operation on a "table" (i.e. a data-file) results in the creation of yet another "temporary table" (another parallel data file), then SQL is unacceptable for generating reports from large data-bases. ACCESS's efficient use of variable-length select-lists appears to be the better way.

At any rate, SQL is not available on any significant sub-set of the types of computers we use today. Unless it becomes a de facto standard, like the EXECUTE command in BASIC, on EVERY machine (UniVerse, Ultimate, GA, etc.) it will remain USELESS to application developers who wish to develop portable software. Which brings us right back to ACCESS. The real purpose of this article is to present the source code to a useful utility, ACCESS.SHELL, which permits one to page ACCESS listings backwards and forwards (and to skip any page instantly).

ACCESS.SHELL is written in conventional BASIC, and should run on all machines capable of handling the F-correlative directive known as "LPV" (load previous value). Although my version of ACCESS.SHELL is tied into a COMMON area, containing terminal-specific definitions such as "dim.on" and "dim.off", the version listed here is a self-contained stand-alone external subroutine module.

Probably the most useful purpose of ACCESS.SHELL lies in the display of cross-reference listings, presented to end-users who type in one or more words from a cross-referenced data-file, thereby narrowing down the list of choices to a

quantity acceptable for manual review. But ACCESS.SHELL can be used for any type of ACCESS listing which meets the criteria defined in the below section "LIMITATIONS".

No attempt will be made here to describe ACCESS.SHELL line-by-line, particularly because there are no line numbers! The program works properly, and is of a "self-commenting" style. The only comments added to "spruce up" the program listing for publication were the definition of passed parameters at the beginning of the program.

EXAMPLES

As a first example, consider the following ACCESS sentence:

```
SORT CUSTOMERS WITH AMOUN.DUE >= "1000" BY-
DSND AMOUNT.DUE ID-SUPP CUST.NUMBER NAME
AMOUNT.DUE PHONE LAST.CONTACTED
```

Now it's obvious the type of listing this would produce on the screen or printer: a fairly straight-backward listing of people who owe us some serious bucks.

To use with ACCESS.SHELL, the following CALL would be made:

```
CALL ACCESS.SHELL('SSELECT CUSTOMERS WITH
AMOUNT . DUE >= "1000" BY-DSND AMOUNT.DUE',
nil, "CUSTOMERS", "CUST.NUMBER NAME
AMOUNT.DUE PHONE LAST.CONTACTED",3,12,1,5,
false, chosen.cust.number)
```

All but the last parameter ("chosen.cust.number") are passed into ACCESS.SHELL. The final parameter, which is always an item-id to the file being listed, in this case "chosen.cust.number", is passed back to the calling program, if the user selects one of the lines presented. Optionally, the user may enter "E" to "Exit without making a choice", in which case "chosen.cust.number" would be returned as null.

Refer to the program listing itself for a detailed description of these parameters. Following the ACCESS.SHELL listing is a test program, SHELL.TEST, which can be used if you supply your own test data. Also included is a front-end program, SHELL-LIST, for invoking ACCESS.SHELL directly from TCL.

Usage Notes

One thing not too terribly elegant about ACCESS.SHELL is the formation of "mini-lists" for each page viewed by each user port. The idea here is optimum efficiency in forward and backward paging - ACCESS.SHELL keeps track of which pages have already been displayed, via the dynamic-array variable "list.built", and can go back to them slightly more quickly.

If for your purposes, this leads to an unacceptable number of lists being stored in the POINTER-FILE, then substitute the following subroutine:

```
*==========
FORM.MINI.LIST:*
*Limits lists-per-port to 2 or 3
mini.list.name = port.number: "*MINI"
current = ((page.number - 1) * records.per.page) + 1
DATA "DE9999", "F"
DATA "ME": records.per.page: "/": port.number: "/": cur-
rent
DATA "FI"
 EXECUTE "EDIT-LIST ": mini.list.name CAPTURING
output
RETURN
*==========
```

This will cause ACCESS.SHELL to use the same list name (e.g. 0005*MINI for port 5) over and over for each page displayed.

LIMITATIONS

There are four limitations to ACCESS.SHELL:

1) One way or the other, a list must be present

before ACCESS.SHELL can start to display any data. This may be either a BASIC variable-list (a dynamic-array attribute-level variable), already prepared in BASIC by the calling program, a select-list generated "on the fly" by ACCESS.SHELL, via a "SELECT" or "GET-LIST", or a select-list activated by the calling program (or generated at TCL)

2) ACCESS.SHELL relies on a fixed number of items per page to be displayed. Therefore, variable-length text-wrapping and varying number of stacked multi-values, must be carefully considered when using with ACCESS.SHELL, and may be dealt with in one of 3 ways:

a) Using the "Tn.m" conversion (e.g. "T1,30" for a name field) to avoid "text wrapping" of attributes, and also avoiding display of multi-valued fields when a maximum number of multi-values cannot be dictated (as in "b" below)

b) Setting the variable "max.lines.per.record" to any number from 2 through 20, which instructs ACCESS.SHELL to reserve vertical depth on the screen for up to "max.lines.per.record" lines per record.

c) Using an EXPLODING-SORT on the file, which clears up all potential multi-value problems,master select list. For instance, a CUSTOMERS file may have a NAME and PHONE field, where PHONE is occasionally multi-valued (but usually not). Simply SSELECT CUSTOMERS BY NAME BY-EXP PHONE to achieve a reliable display with ACCESS.SHELL (see example in SHELL.TEST as multi-value numbers become built right into the named EXPLODING. SORT.TEST).

Although it would certainly be possible to enhance ACCESS.SHELL to accommodate this "wrapping" limitation performance would go down the tube, as BASIC would have to first read in and track the "wrapping depth" and/or "multi-value count", for every record. This feature is best saved for the systems manufacturers, who by absorbing ACCESS.SHELL into assembly language (i.e. directly into the ACCESS assembly modes) could achieve satisfactory performance of this concept.

3) The ME (merge) editor command is used to merge the proper number of item-ids from the "master select list" into individual "page-level mini-lists". However, the ME command malfunctions when one exceeds 32676 items in the master select list, and ME is unable to retrieve anything beyond 32767. This should rarely present a problem, as no same user is going to want to page through nearly 2000 pages.

4) Dictionary items referenced must not have "stacked" column headings (stored in AMC 3) deeper than that indicated by the parameter "access.heading.lines" (which should be set one higher than the actual number of heading lines, as ACCESS always puts our a blank line as part of the heading).

To receive a copy of ACCESS.SHELL, send a formatted (500-byte block-size) standard density diskette to the address below, with a self-addressed stamped return envelope. You will be sent two versions of ACCESS.SHELL, one with its meaningful alphabetic statement labels, one without (for comp[ilation by compilers unable to handle meaningful labels), and also a copy of SHELL.TEST and SHELL.LIST. ACCESS.SHELL is hereby considered public-domain software, and may not be directly sold or re-sold. It may, however, be incorporated into, and sold along with any application software, or incorporated into systems at the assembly-level, by the manufacturers. Send requests to Dave Ramette, ACE Software, PO Box 442, Laguna Beach, CA 92652.

Source Code follows on next page:

```
ACCESS.SHELL
SUBROUTINE ACCESS.SHELL(access.select, passed.list, file.name, output.cr
iteria, num.heading.lines, records.per.page, max.lines.per.record, begin
.line, inquiry.only, chosen.item.id)
*
*============
*
*    Surround ACCESS with user-friendly BASIC shell.
*      Worldwide public-domain software.
*      Author: Dave Ramette (ACE Software), March, 1990.
*
*============
*
*    Definition of passed parameters:
*
*    If both of the first two parameters are passed as "nil",
*    then there MUST be a select-list active (fetched by the
*    calling program via EXECUTING a SELECT, SSELECT, GET-LIST,
*    QSELECT, etc.).
*
*     access.select:  Any valid TCL list-retrieval statement
*                     (SELECT, SSELECT including WITH, BY, USING
*                     clauses, GET-LIST, QSELECT, etc.).  Takes
*                     precedence over "passed.list".
*
*       passed.list:  Dynamic-array (attribute-level) list of item-
*                     ids prepared by calling program.  Referenced
*                     only if "access.select" is passed as null.
*
*         file.name:  Name of data file being listed.
*
*  output.criteria:  The ACCESS "output-criteria" clause, i.e.
*                     a list of field-names to display.  May include
*                     ACCESS directive COL-HDR-SUPP to increase
*                     maximum "records.per.page" (below) from 18 to 20.
*                     Since ID-SUPP is automatically supplied,
*                     an actual item-id field-name must be specified
*                     here if display of item-ids is desired.
*
* num.heading.lines:  The total number of heading lines, which should
*                     equal the maximum number of "stacked column
*                     headings" defined in AMC 3 of all fields refer-
*                     enced by "output.criteria", plus one (1) for the
*                     extra blank heading line supplied by ACCESS.
*                     Set to 2 for displays without "stacked column
*                     headings".  Set to 0 if COL-HDR-SUPP is used.
*
* records.per.page:  Number of items to display on each page.
*                     Minimum:  1.
*                     Maximum: 18 without COL-HDR-SUPP,
```

```
*                                   20 with COL-HDR-SUPP.
*
*
* max.lines.per.record:
*                       The maximum "wrapping depth" and/or "multi-
*                       value count expected to be displayed.  The worst
*                       that can happen, if ACCESS ends up displaying more
*                       than that specified, would be that individual
*                       pages may partially "scroll off the screen" before
*                       ACCESS.SHELL presents the display prompt.
*
*               NOTE:   The total number of HEADING/DATA lines
*                       (num.heading.lines + (records.per.page x
*                       max.lines.per.record)) should not exceed 20
*                       lines on standard terminals (i.e. terminals
*                       with 24 lines:  0 through 23).
*
*         begin.line:   Beginning line at which to present display,
*                       permitting "split-screen" display (as clear-
*                       lines are used to clear area, not clear-screen).
*                       If passed as 0, will be changed to 1.
*
*               NOTE:   The total number of OVERALL lines (begin.line +
*                       num.heading.lines + (records.per.page x max.lines.
*                       per.record)) should not exceed 21 lines on standard

*                       terminals (ACCESS.SHELL reserves line 0 for the
*                       purpose of backing up the cursor one line for
*                       proper display, since ACCESS causes the cursor to
*                       "jump forward" one line; ACCESS.SHELL uses 2 lines
*                       at the bottom for the display prompt).
*
*       inquiry.only:   If set to true (1), then the prompt display will
*                       not show the "Enter line #" option.
*
*    chosen.item.id:    Item-id of record chosen by user.  Returned as
*                       null if user enters "E" to "Exit without making
*                       a choice".  Calling program takes action based
*                       on the result returned in this variable.
*                       Not generally applicable (but still available)
*                       when using "inquiry.only" mode.
*
*============
EQU false TO 0, true TO 1, otherwise TO 1, nil TO ""
EQU attr.mark TO CHAR(254), bell TO CHAR(7), space TO " "
*============
ACCESS.SHELL:*
 GOSUB INITIALIZE
 dropout = false
 GOSUB SELECT.LIST
```

```
 IF NOT(dropout) THEN GOSUB DISPLAY.LOOP
 EXECUTE "TERM ,,,,2" CAPTURING output :* reset formfeed delay
RETURN
*============
INITIALIZE:*
 PROMPT nil
 who = OCONV("", "U50BB")
 port.number = FIELD(who, " ", 1) "R%4"
 clear.line = @(-4)
 dim.on = @(-7)
 dim.off = @(-8)
 *
 OPEN "POINTER-FILE" TO pointer.file ELSE STOP "201", "POINTER-FILE"
 OPEN "DICT", file.name TO dict.data.file ELSE STOP "201", "DICT ": file
.name
 *
 IF (inquiry.only) THEN line.number.string = nil ELSE line.number.string
 = "Line# or "
 line.number      = "S"
 line.number<2>   = "9998"
 line.number<3>   = "Line#"
 line.number<9>   = "R"
 line.number<10> = "5"
 *
 IF (begin.line <= 0) THEN begin.line = 1
 IF (max.lines.per.record <= 0) THEN max.lines.per.record = 1
 end.line = (begin.line + num.heading.lines + (records.per.page * max.l
ines.per.record))
 list.built = nil
 chosen.item.id = nil
 *
RETURN
*============
SELECT.LIST:*
 BEGIN CASE
  CASE (access.select = nil AND passed.list = nil)
   GOSUB VERIFY.SELECT
  CASE (access.select # nil)
   EXECUTE access.select RETURNING errors CAPTURING output
   GOSUB VERIFY.SELECT
  CASE (otherwise)
   items.selected = DCOUNT(passed.list, attr.mark)
   IF (items.selected = 0) THEN GOSUB NO.ITEMS.MESSAGE ELSE
    WRITE passed.list ON pointer.file, "?": port.number
    EXECUTE "QSELECT POINTER-FILE ": "?": port.number CAPTURING output
    GOSUB VERIFY.SELECT
   END
 END CASE
 max.page = INT(items.selected / records.per.page + .9999)
RETURN
```

```
*============
VERIFY.SELECT:*
 *
 * VERIFY.SELECT assumes that the system will output message 240
 * when a SAVE-LIST of "no items" is executed, and a 404 when a
 * GET-LIST of "some items" is executed.  This prevents us from
 * having to DELETE-LIST the port's master-list each time ACCESS.
 * SHELL is called (calling program may want to reference this list).
 *
 * VERIFY   MESSAGES   240   AND   404   IN   "ERRMSG"   FILE !!!
 *
 EXECUTE "SAVE-LIST ": port.number RETURNING errors CAPTURING output
 LOCATE("240", errors; dummy) THEN GOSUB NO.ITEMS.MESSAGE ELSE dropout =
 false
 IF NOT(dropout) THEN
  EXECUTE "TERM ,,,,0" CAPTURING output ;* avoid ACCESS clear-screen
  EXECUTE "GET-LIST ": port.number RETURNING errors CAPTURING output
  items.selected = FIELD(output, space, 2)
  LOCATE("404", errors; dummy) THEN dropout = false ELSE GOSUB NO.ITEMS.
MESSAGE
 END
RETURN
*============
NO.ITEMS.MESSAGE:*
 PRINT @(0, begin.line)
 PRINT "No ": file.name: " on file...press RETURN...":
 INPUT temp:
 items.selected = 0
 dropout = true
RETURN
*============
DISPLAY.LOOP:*
 message2 = dim.on: \F)irst  L)ast  Pn)Skip to "n"  An)Ahead "n"  Bn)Bac
k "n" pages  E)xit ?\: dim.off
 all.done = false
 page.number = 1
 exec = "LIST ": file.name: " HDR-SUPP ID-SUPP LINE*": port.number
 exec = exec: " ": output.criteria
 faulty.command = false
 LOOP
  IF NOT(faulty.command) THEN GOSUB FORM.MINI.LIST
 UNTIL (all.done) DO
  IF (faulty.command) THEN
   PRINT bell:
   faulty.command = false
  END ELSE
   GOSUB CLEAR.WINDOW
   line.number<7> = "F:LPV:C": (page.number - 1) * records.per.page: ":+
"
   WRITE line.number ON dict.data.file, "LINE*": port.number
```

```
    EXECUTE "GET-LIST ": mini.list.name CAPTURING output
    EXECUTE exec RETURNING temp1
   END
   GOSUB COMMAND.LINE
  REPEAT
 RETURN
 *============
 FORM.MINI.LIST:*
  mini.list.name = page.number: "*": port.number
  IF NOT(list.built<page.number>) THEN
   current = ((page.number - 1) * records.per.page) + 1
   DATA "DE9999", "F"
   DATA "ME": records.per.page: "/": port.number: "/": current
   DATA "FI"
   EXECUTE "EDIT-LIST ": mini.list.name CAPTURING output
   list.built<page.number> = true
  END
 RETURN
 *============
 ALTERNATE.FORM.MINI.LIST:*
  *Limits lists-per-port to 2 or 3
  mini.list.name = port.number: "*MINI"
  current = ((page.number - 1) * records.per.page) + 1
  DATA "DE9999", "F"
  DATA "ME": records.per.page: "/": port.number: "/": current
  DATA "FI"
  EXECUTE "EDIT-LIST ": mini.list.name CAPTURING output
 RETURN
 *============
 CLEAR.WINDOW:*
  FOR line = begin.line TO end.line
   PRINT @(0,line): clear.line:
  NEXT line
  PRINT @(0, begin.line-1):
 RETURN
 *============
 COMMAND.LINE:*
  message1 = dim.on: items.selected: space: file.name: "...Page ": page.n
umber: " of ": max.page: "...Enter ": line.number.string: "command:  RET
URN=Ahead"
  PRINT @(0, end.line+1): message1: clear.line:
  PRINT @(0, end.line+2): message2: clear.line:
  INPUT command,6:
  command = OCONV(command, "MCU")
  command.1 = command[1,1]
  BEGIN CASE
   CASE (command   = nil) ; GOSUB AHEAD.PAGE
   CASE (command   = "F") ; GOSUB FIRST.PAGE
   CASE (command   = "L") ; GOSUB LAST.PAGE
   CASE (command.1 = "P") ; GOSUB SKIP.TO.PAGE
```

```
   CASE (command.1 = "A") : GOSUB JUMP.AHEAD.PAGES
   CASE (command.1 = "B") : GOSUB JUMP.BACK.PAGES
   CASE (command   = "E") : all.done = true
   CASE (NUM(command) AND command # nil) : GOSUB FETCH.SELECTED.ID
   CASE (otherwise)       : faulty.command = true
 END CASE
RETURN
*============
AHEAD.PAGE:*
 IF (page.number < max.page) THEN page.number = (page.number + 1) ELSE f
aulty.command = true
RETURN
*============
FIRST.PAGE:*
 page.number = 1
RETURN
*============
LAST.PAGE:*
 page.number = max.page
RETURN
*============
SKIP.TO.PAGE:*
 skip = command[2,99]
 IF NUM(skip) THEN IF (skip >= 1 AND skip <= max.page) THEN page.number
= skip ELSE faulty.command = true
RETURN
*============
JUMP.AHEAD.PAGES:*
 ahead = command[2,99]
 IF (ahead = nil) THEN ahead = 1
 IF (NUM(ahead) AND ahead >= 1) THEN
  IF (page.number + ahead <= max.page) THEN jump.ahead = ahead ELSE
   IF (ahead < max.page) THEN jump.ahead = (ahead - max.page) ELSE
    faulty.command = true
   END
  END
 END ELSE faulty.command = true
 IF NOT(faulty.command) THEN page.number = (page.number + jump.ahead)
RETURN
*============
JUMP.BACK.PAGES:*
 back = command[2,99]
 IF (back = nil) THEN back = 1
 IF (NUM(back) AND back >= 1) THEN
  IF (page.number - back >= 1) THEN jump.back = back ELSE
   IF (back < max.page) THEN jump.back = (back - max.page) ELSE
    faulty.command = true
   END
  END
 END ELSE faulty.command = true
```

```
 IF NOT(faulty.command) THEN page.number = (page.number - jump.back)
RETURN
*============
FETCH.SELECTED.ID:*
 IF (command > 0 AND command <= items.selected) THEN
  * Assume system editor can go to a line without "G" prefix:
  DATA command, "EX"
  EXECUTE "EDIT-LIST ": port.number CAPTURING output
  chosen.item.id = OCONV(output<3,1>, "G1 99")
  all.done = true
 END ELSE faulty.command = true
RETURN
*============
```

```
SHELL.LIST
*
*============
*
*    Front-end program for invoking ACCESS.SHELL from TCL.
*      Worldwide public-domain software.
*      Author: Dave Ramette (ACE Software), March, 1990.
*
*    SHELL.LIST transforms an ACCESS-like statement into parameters
*    necessary to call ACCESS.SHELL.  If WITH, BY, BY-EXP, and/or USING
*    clauses are required, first perform them with conventional ACCESS
*    (SELECT or SSELECT), thereby generating an active select-list.
*    Then, invoke SHELL.LIST with the simple format:
*
*    >SHELL-LIST file.name {output.criteria} {(options)}
*
*    All alphabetic options (A-Z) will be passed directly to ACCESS.
*    A numeric "option" (1 through 20), if specified, will set the
*    parameter "max.lines.per.record" (defaults to 1).
*
*    This program requires that a small PROC be placed in the
*    master dictionary (MD file) in order for PROCREAD to work:
*
*        SHELL-LIST
*    001 PQ
*    002 HSHELL.LIST (E
*
*============
EQU false TO 0, true TO 1, nil TO "", space TO " "
*------------
SHELL.LIST:*
 PROCREAD input.buffer ELSE STOP "PROCREAD input.buffer"
 verb              = FIELD(input.buffer, space, 1)
```

```
 file.name        = FIELD(input.buffer, space, 2)
 output.criteria = OCONV(input.buffer, "G2 99")
 *------------
 word.count = DCOUNT(output.criteria, space)
 last.word = FIELD(output.criteria, space, word.count)
 IF (last.word[1,1] = "(") THEN
  options = last.word
  output.criteria = OCONV(output.criteria, "GO ": word.count-1)
  max.lines.per.record = OCONV(options, "MCN")
  IF (max.lines.per.record = nil) THEN max.lines.per.record = 1
  options = OCONV(options, "MCA")
  IF (options # nil) THEN options = " (": options
 END ELSE
  options = nil
  max.lines.per.record = 1
 END
 *------------
 *...Maximum with COL-HDR-SUPP on standard terminals:  0, 20, 1, 1
 *...Maximum without COL-HDR-SUPP on standard terminals:  2, 18, 1, 1
 IF (INDEX(output.criteria, "COL-HDR-SUPP", 1) OR INDEX(options, "C", 1)
) THEN
  num.heading.lines = 0
  max.records = 20
 END ELSE
  num.heading.lines = 2
  max.records = 18
 END
 records.per.page = INT(max.records / max.lines.per.record)
 *------------
 IF SYSTEM(11) THEN access.select = nil ELSE
  access.select = "SELECT ":file.name
 END
 CALL ACCESS.SHELL(access.select, "", file.name, output.criteria:options
, num.heading.lines, records.per.page, max.lines.per.record, 1, true, ""
)
STOP
*============
```

```
SHELL.TEST
*
*============
*
*    Program to test all types of ACCESS.SHELL calls
*      Worldwide public-domain software.
*      Author: Dave Ramette (ACE Software), March, 1990.
*
*    To use, define your own CUSTOMERS file with dictionary items:
```

```
*
*    CUST.NUMBER        A^0^Customer]number^^^^^^L^8
*    NAME               A^1^Name^^^^T1,30^MCT^L^30
*    AMOUNT.DUE         A^2^Amount]due^^^^MR2,$^^R^10
*    PHONE              A^3^Phone^^^^^^L^16
*    LAST.CONTACTED     A^4^Last]contacted^^^^D2/^^L^8
*
*============
EQU false TO 0, true TO 1, nil TO ""
*============
SHELL.TEST:*
 clear.screen = @(-1)
 show.multi.value = false
 GOSUB SELECT.LIST.ACTIVE.TEST
 GOSUB SSELECT.TEST
 GOSUB GET.LIST.TEST
 GOSUB PASSED.LIST.TEST
 show.multi.value = true
 GOSUB EXPLODING.SORT.TEST
STOP
*============
SELECT.LIST.ACTIVE.TEST:*
 test.name = "SELECT.LIST.ACTIVE.TEST" : GOSUB BARS.EVERYWHERE
 exec = 'SSELECT CUSTOMERS WITH AMOUNT.DUE >= "1000" BY-DSND AMOUNT.DUE'

 EXECUTE exec CAPTURING output
 CALL ACCESS.SHELL(nil, nil, "CUSTOMERS", "CUST.NUMBER NAME AMOUNT.DUE P
HONE LAST.CONTACTED", 3, 12, 1, 2, false, chosen.cust.number)
 GOSUB SHOW.CHOSEN
RETURN
*============
SSELECT.TEST:*
  !
 OPEN "CUSTOMERS" TO customers.file ELSE STOP "201", "CUSTOMERS"
  !
 test.name = "SSELECT.TEST" : GOSUB BARS.EVERYWHERE
 CALL ACCESS.SHELL('SSELECT CUSTOMERS WITH AMOUNT.DUE >= "1000" BY-DSND
AMOUNT.DUE', nil, "CUSTOMERS", "CUST.NUMBER NAME AMOUNT.DUE PHONE LAST.C
ONTACTED", 3, 12, 1, 4, false, chosen.cust.number)
 IF (chosen.cust.number # nil) THEN
  READU customers.item FROM customers.file, chosen.cust.number THEN
   !
  GOSUB SHOW.CHOSEN
   !
  END
 END
  !
RETURN
*============
PASSED.LIST.TEST:*
```

```
 EXECUTE 'SSELECT CUSTOMERS WITH NAME >= "A" AND <= "M" BY NAME'
 passed.list = nil
 done = false
 LOOP
  READNEXT id ELSE done = true
 UNTIL (done) DO
  passed.list<-1> = id
 REPEAT
 test.name = "PASSED.LIST.TEST" : GOSUB BARS.EVERYWHERE
 CALL ACCESS.SHELL(nil, passed.list, "CUSTOMERS", "CUST.NUMBER NAME AMOU
NT.DUE PHONE LAST.CONTACTED", 3, 12, 1, 6, false, chosen.cust.number)
 GOSUB SHOW.CHOSEN
RETURN
*============
GET.LIST.TEST:*
 test.name = "GET.LIST.TEST" : GOSUB BARS.EVERYWHERE
 CALL ACCESS.SHELL("GET-LIST GET.LIST.TEST", nil, "CUSTOMERS", "CUST.NUM
BER NAME AMOUNT.DUE PHONE LAST.CONTACTED", 3, 10, 1, 8, false, chosen.cu
st.number)
 GOSUB SHOW.CHOSEN
RETURN
*============
EXPLODING.SORT.TEST:*
 test.name = "EXPLODING.SORT.TEST" : GOSUB BARS.EVERYWHERE
 CALL ACCESS.SHELL("SSELECT CUSTOMERS BY NAME BY-EXP PHONE", nil, "CUSTO
MERS", "CUST.NUMBER NAME AMOUNT.DUE PHONE LAST.CONTACTED", 3, 4, 1, 6, f
alse, chosen.cust.number)
 GOSUB SHOW.CHOSEN
RETURN
*============
BARS.EVERYWHERE:*
 PRINT clear.screen: test.name:
 PRINT STR("!", 1919-LEN(test.name)):
RETURN
*============
SHOW.CHOSEN:*
 * NOTE:  The EXECUTE CAPTURING clause converts value-
 * marks to right-brackets, ]'s, so for exploding-sorts, extract
 * elements with the FIELD command:
 PRINT :  PRINT
 PRINT "chosen cust-number: ": FIELD(chosen.cust.number, ']', 1):
 IF (show.multi.value) THEN
  PRINT : PRINT "chosen multi-value: ": FIELD(chosen.cust.number, ']', 2
):
 END
 INPUT asdf
RETURN
```

High Performance PC Products for:

Multi-User PICK/DOS/UNIX/XENIX/NOVELL Solutions

New

KRAFT ENTERPRISES

QIK-XL2

9 Track 1/2" Tape Software

For PC, XT, AT, 386, PS/2

with Controller Starting at

$1395.00*

** Also includes DOS Software*

New

Overland Data Inc.

9 Track Tape Controllers

• TX 8 for PC, XT, AT, 386

with drivers for

MS-DOS/PC-DOS, XENIX, UNIX

•XL/2 for IBM PS/2

New

DPT Distributed Processing Technology

CACHING and MIRRORING

Speed and Security Now

in One Package with

SmartCache

.5ms access time - Disk fault-tolerance

Supports all PC/AT operating systems

CALL TODAY FOR OUR CATALOG —DEALER PRICING AVAILABLE

— SYSTEMS —

Capricom Data, Everex,
Compaq, Acer

—PERIPHERALS —

CRT's, Printers, Modems,
Tape Backups, Optical Storage

TAPE DRIVES

New and Refurbished

CIPHER, KENNEDY, QUALSTAR

Prices as low as $2000.00

SOLUTIONS:

We'll help you find the answers

Amet
Bristol
Digiboard
DPT (Distributed Processing Technology)
Kraft Enterprises
Overland Data
Pick Systems

Nationwide Service by: NCE

SOFTWARE: Datamax Accounting, Payroll, DMS Distribution, JM Manufacturing
Graphics • Spreadsheets • Autosizing • TCL Stackers • Operating Systems • Upgrades
Communications • Connectivity • Business Applications

SUPPLIES: Custom Forms, Paper, Ribbons, Tape Reels, Data Cartridges

COMPLETE SERVICE FOR ALL YOUR DATA PROCESSING NEEDS

CAPRICORN DATA, INC.

1285 Stone Dr., Ste. 101
San Marcos, CA 92069

(619) 471-9307
Fax: 619-471-1018

System Security:
Business Insurance for your Company

by Jeff Stutz, President Apscore International

In life there is a saying that goes "What you don't know can't hurt you", but when it comes to insuring the integrity and security of your information systems, "What you don't know can kill you!" Businesses today cannot operate without their computers. Every decision is based directly or indirectly on information provided by these machines, and the information stored on them is a key corporate asset. Yet, in spite of all the current headlines, many, perhaps most systems are not secure from a variety of threats.

1) **Theft of Information.** Information of strategic importance is frequently easy to copy and steal. All that may be necessary is an unauthorized print-out or unauthorized access to a terminal. Since the original information is not altered but simply copied, this form of theft frequently goes undetected.

2) **Vandalism.** Unauthorized tampering with data or programs can result in computer problems leading to downtime, loss of information or considerable repair costs.

3) **Tampering for personal gain.** Many businesses ship valuable products, account for vast sums of cash or produce payroll checks using their computers. An unsecured system represents an open invitation for theft or misappropriation of corporate assets.

4) **Hacking.** For many people, accessing restricted computers and their data is a game. The threat can come from any direction and, in most cases, cannot easily be anticipated.

What can a company do to protect themselves from a breech in system security?

The System Security Checklist

The following is a checklist of requirements that must be met to insure the most basic levels of system security. Use it as a self-audit to help determine your company's level of security risk.

1) Maintain individual passwords for each user, and be able to change them frequently.

2) Do not allow users to re-use changed passwords.

3) Monitor all failed attempts to access the computer and automatically lock-out the user Id and terminal when failed attempts have exceeded a certain number.

4) Keep a complete audit-trail of access to the system, its programs and files.

5) Have the ability to keep a selective audit-trail of changes to data.

6) Automatically lock terminals that are logged-on but which remain inactive.

7) Control and limit access to operating system utilities.

8) Prevent observation of user IDs and passwords.

9) Maintain an audit trail of user passwords.

10) Easily change user status from active (lon on permitted) to inactive (log on denied) and back again.

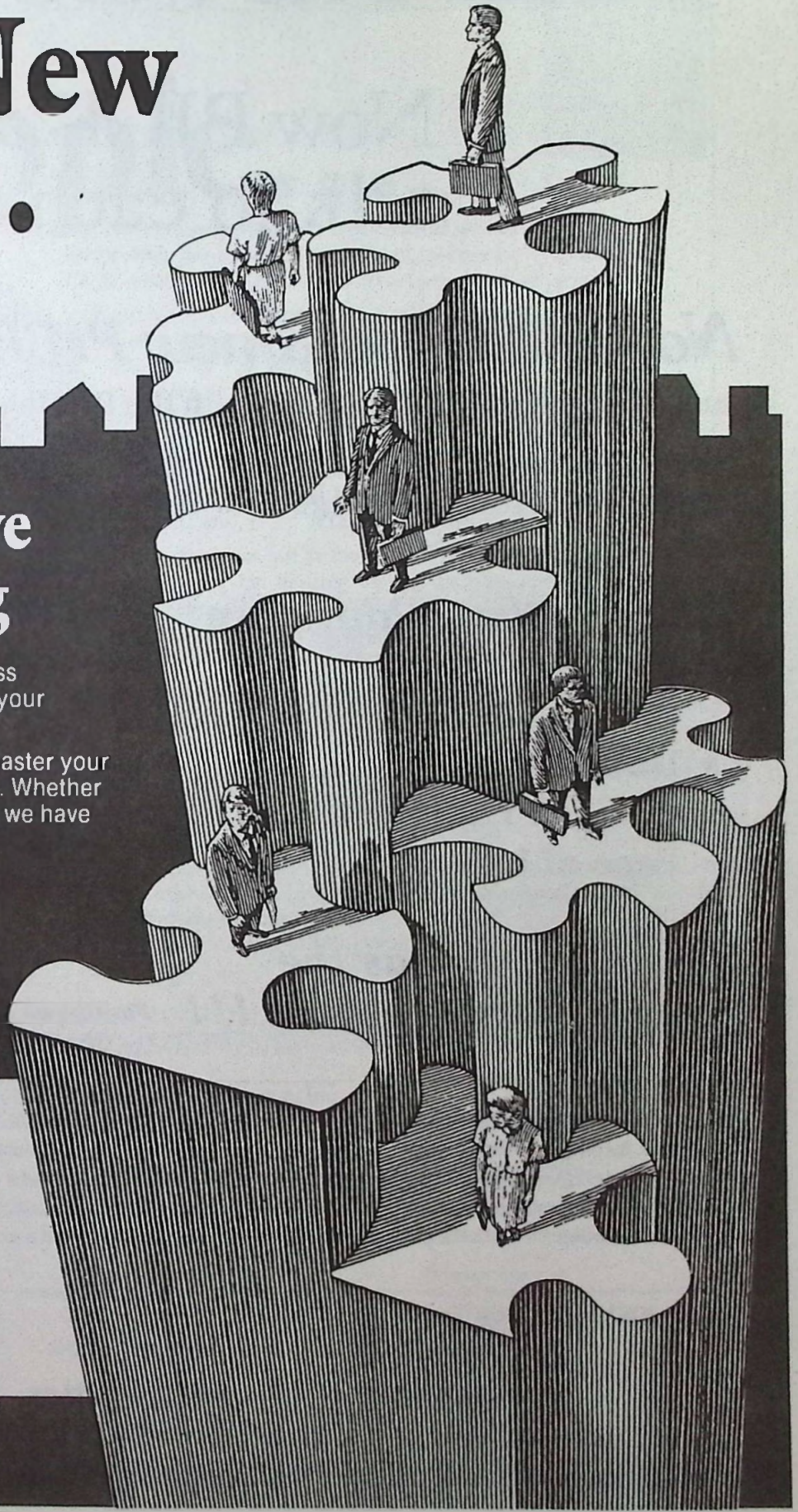
11) Restrict specific programs to specific terminals.

System security is just as critical to your company as business insurance. Without it you are risking your company's ability to do business and service your customers. The key question is, can your company afford to be without a secure computer environment?

Jeff Stutz is President of Apscore Internationa, Inc. a leader in security systems and security consulting for PICK and PICK/UNIX systems.

If your system is not secure or if you are concerned about system security, please call 1-800-3-CUE-BIC for more information.

Reach New Heights.



Comprehensive Pick® Training

In today's computer dependent business world, your success can depend upon your computer skills.

We can provide you with the skills to master your system and the information it manages. Whether you are a beginner or are experienced, we have the classes for you — in over 20 cities nationwide. We can also train you at your location.

By helping our clients become #1 in their industry, we have become #1 in our industry. Let us help you reach new heights in productivity and effectiveness while minimizing costly expenses and downtime.

Comprehensive Computer Services Inc.

The Leader in PICK Education sm

Dept. 777

P.O. Box 296, East Islip, NY 11730

FOR INFORMATION OR A FREE BROCHURE

CALL (516) 581-7919

FAX (516) 277-2624

The word is out...

Now PICK® and DOS® talk to each other.

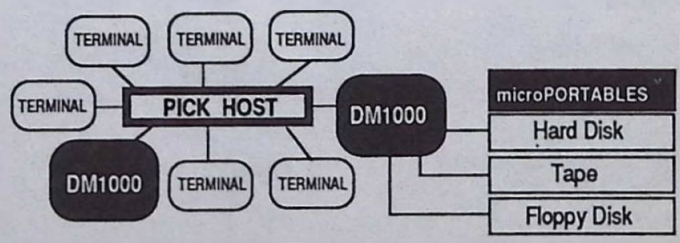
No need to get a new PICK system . . .

Just enhance the one you have with the DMI Intelligent Workstation which can function as a WordPerfect Workstation, Lotus Workstation or Human Resource Management Workstation – and many more.

If these three things are important to you . . .

- **Data Integrity**
- **One Stop Support**
- **Ease of Use**

. . . then DMI has the only solution in town !!!



The DM 1000 brings the Pick and DOS worlds together in a uniform, small, quiet and affordable unit.

Finally, an intelligent workstation for the PICK environment is here.

Current Configurations:

DM1000 Intelligent Workstation (Diskless)
DM1020 Intelligent Workstation (with 20MB ext. portable disk)
DM1044 Intelligent Workstation (with 44MB ext. portable disk)
DM1104 Intelligent Workstation (with 104MB ext. portable disk)

All DMI Workstations come complete with the necessary hardware and software for immediate use. MS/DOS, Word Perfect Executive application, and communication software are all installed on the DM1000.

Prices start @ $1995.

Dealers/OEMS - Sign Up Now.

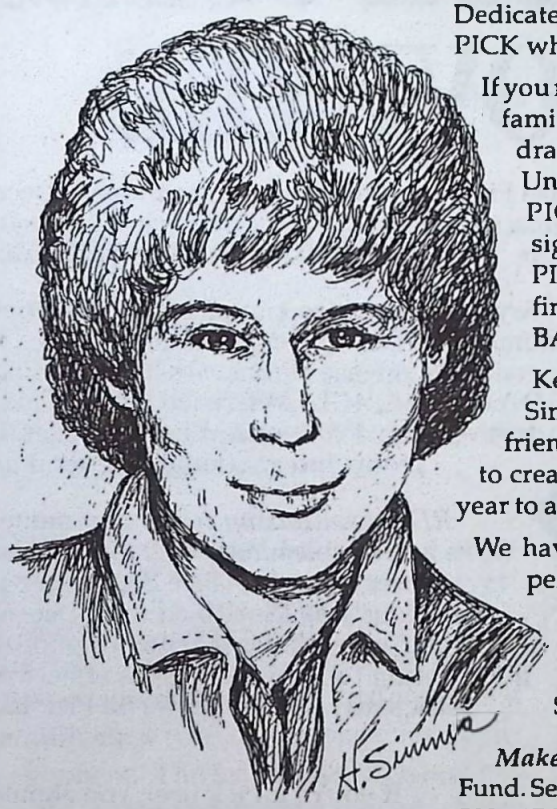
You think of the possibilities. We'll make it happen.

dmi®

1 Hughes Irvine, CA 92718 • (714) 583-1800 • FAX (714) 583-9381

People...

the most significant asset of the PICK marketplace. Dedicated, hard-working, resilient people have made PICK what it is today. People like these keep us going.



If you recognize this man as a familiar face in the PICK family, you're right. This is a sketch of Ken Simms, drawn by his father. Ken was a student at the University of California at Irvine in 1974 (where PICK began) and became one of the most significant contributors to the development of the PICK Operating System. He was PICK System's first employee and went on to develop PICK BASIC and, later, WIZARD Software.

Ken died last fall at the age of 38. We honor Ken Simms for his genius, vision, and tenacity. His friends, family and colleagues have joined together to create a memorial scholarship to be awarded each year to a deserving U.C.I. student of Computer Science.

We have achieved our initial goal of $10,000, and a permanent fund has been established. Each year approximately six percent of the scholarship fund will be awarded—forever. We'd like you to join us and invest in the future. The Ken Simms Fund needs people like you.

Make checks payable to: UCI Foundation - Ken Simms Fund. Send checks to: Laguna Software & Consulting, Inc., 580-117 Broadway, Laguna Beach, CA 92651.
For more information, call Bill Wulff or Steve Lambert at 714-494-1092.

CURRENT LIST OF CONTRIBUTORS TO KEN SIMMS FUND

Fujitsu Microsystems of America, Inc.
PICK Systems

Laguna Software & Consulting, Inc.
Thurman Marketing Services, Inc.
JES & Associates, Inc.

Kenneth Hoppe
OTRA Clearing, Inc.
Stratus Computer, PICK Division
Linda Rodino

The Consultant Company
JET Software
Jack Cory

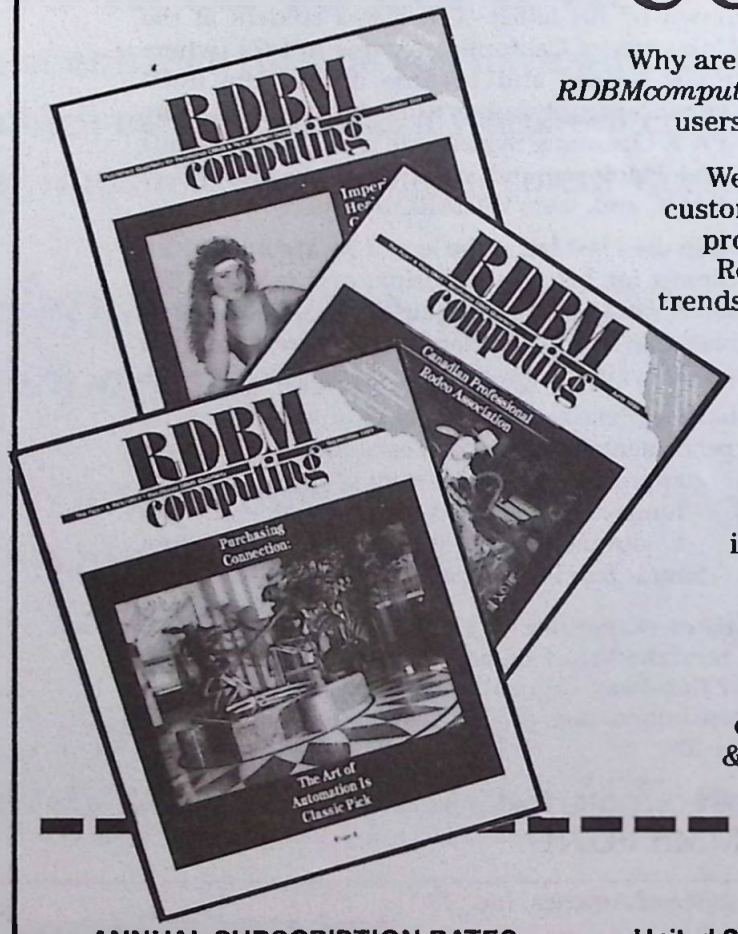
Mr. & Mrs. Herman H. Simms
BONNECO
Diversified Data Corporation
Jerome & Hazel Tobis
Michael F. Wisbard
Peterson Research
System Builder Technologies
Wayne Heldt
AccuSoft Enterprises
Interactive Systems
Bertha Green

David & Deborah Raber
Gardner, Saunders & Associates
Rachel E. Dolan
Edith Wolock
Abest Computer Systems, Inc.
Jack Berry
General Automation
Joy Butler
Karen Schaltenbrand
Bob Gutherie & Gemma Connors
Atkin/Jones Computer Service, Inc.

Computer Clinic
Bayada Nurses
`Gar Marz
D.O.X. INC.
Specialty Software
John & Dorothy Barry
Millington & Associates
Philip Kramer
Michael Castronova
J. & R. Leason
Kathleen Goss

RDBM computing

WE'VE GOT PICK USERS COVERED



Why are Pick users always our cover story? Because *RDBMcomputing* is written for Pick & Pick/UNIX system users. PC users...mini users...mainframe users...

We've got lots more in addition to our popular customer profile features. Technical articles. New product & release summaries. Indepth Special Reports (i.e., 4GLs, reference library). Market trends & opinion. Product & vendor profiles. User group, international & general news.

RDBMcomputing is our new name but we've been publishing a Pick user magazine every quarter since September 1984 (previously as *PickWorld* until December 88). Our magazine is devoted to publishing information to help users more effectively & efficiently use the powerful Pick RDBM environment.

If you're a Pick user, you should be reading *RDBMcomputing* from cover to cover. Send in your subscription form today & we'll send you our latest issue immediately.

ANNUAL SUBSCRIPTION RATES
(4 issues)

	1 Year/2 Years
United States	**$25/$45**
Canada/Mexico	**$35 USD/$65 USD***
All Other Countries	**$50 USD/$95 USD***

Must be U.S. dollars, international money order, or U.S. bank check.

(Please print or attach your business card.)

Name: _____ Title: _____

Company Name: _____ Telephone: (____) _____

Mailing Address: _____

City: _____ State: _____ Country: _____ Zip: _____

Computer system(s): _____

Please mail with check to publisher or distributor. (No invoice will be sent.)

Publisher
RDBMcomputing
P.O. Box 9539
Fountain Valley, CA 92728-9539
714/839-6708

U.S. Distributor
GARDNER SAUNDERS & ASSOCIATES
1650 S. Amphlett Blvd., Ste. 317
San Mateo, CA 94402
415/341-PICK

International Distributor (Europe, Middle East & Africa)
ALLM Systems & Marketing
21 Beechcroft Road
Bushey, Herts, England WD2 2JU
(0923) 30150

E D +
a Full Screen Editor for PICK Users

Reviewed by Sheldon Markesteyn-Smith, Consultant

"Real programmers write code in Fortran. . . " or so you've been told. As a PICK programmer you know better but that's not the point of this review. The point is, regardless of your level of programming expertise, ED + (pronounced E-D-Plus) will make you a faster, more effective, and in some cases even a better programmer!

-What is ED + -

ED + is a "non-midal", full-screen editor developed by a programmer specifically for programmers. This one fact gives ED + a distinct advantage over any word-processor of line-oriented editor that is available in the Pick (tm) marketplace for applications development. The author, Kevin King, has done a masterful job and provided a tool which has been sorely needed for quite sometime. Overall, ED + is one of the most powerful and useful tools that I've encountered in the 14 years that I've worked with the Pick Operating System. If I've gotten your interest, read on to learn about some of the many ED + features that make it so powerful.

-Installing ED + -

ED + is provided on suitable media for your system and requires only a few simple steps for installation. Most of the installation tasks are handled by an automated installation process, so there's very little chance of a bad install. Several popular terminal definitions are provided, but of course this list is not all-inclusive. This is generally not a problem as defining terminals is done quite easily using the DEFINE-TERMDEF and COMPILE-TERMDEF verbs. With this flexibility, supporting any terminal is possible.

-Starting ED + -

ED + is started in a similar fashion to the Pick System Editor with the simple command string:

ED + filename itemname {itemname...} {(options)}

Substituting a "?" for itemname will produce a screen of items available in the current filename. Appending a valid "WITH..." clause to the "?" will show only those items matching the "WITH..." clause. You may then use the cursor keys to scroll the highlighted market to the appropriate item. Several command-line options are provided, most notably, (C)ase Sensitivity, (T)imed Backup, and (R)ecover from last timed backup (when was the last time you typed 'FD' and said "Oh Noooo!" or maybe something a bit stronger?). Once you've entered <RTN> you've got access to some of the best editing features around and YOU have total control over how they're invoked... Now that's flexibility!

-ED + Features-

Control is provided via extensive use of function keys and cursor control keys. If you don't have them, not to worry, ED + is fully configurable on a per port basis to suit individual requirements.

BACK ISSUES OF PICK® USER DIGEST

March / April 1989 1	**Articles:** Barcodes Enhance Profitability• Everything Old is New Again • Using Check Digits to Reduce Input Errors • Structured Programming • Why move to Advanced Pick? • The Correlative • A Simple C.Itoh printer driver • Reality and the Relational Model • Secrets of the PSYM File • A BASIC Precompiler (Pt 2)	January/ February 1990 6	**Articles:** SQL - The Logical Standard? • How EMS Works • PICK is not a Relational DBMS • A Workstation for the PICK Environment • Undocumented and Overlooked Features of R83 Release 3.0 • Technical Tips • Toolbox Utilities • Helpful Hints for the Novice PICK User
May 1989 2	**Articles:** Extended Terminal Control • UniVerse - in Support of PICK Applications • System Security for the Novice • To PROC or not to PROC • Correlatives - What? Why? How? Part 2 • Nested IF Statements • MAKELIST - A powerful LIST management program • Barcodes enhance profitability Part 2 • TOOLBOX - Capturing TCL and ACCESS Output • Helpful Hints for the Small Business	March/ April 1990 7	**Articles:** Access Shell - A Paging Utility for Access Users • Why PCs Do Not Run As Well As Minis • System Security: Business Insurance for Your Company • ED+ a Full Screen Editor for Pick Systems • File Sizing Faceoff • TOOLBOX - Assigning and Controlling Sequential Keys • Conversion Code for the Pick User
July 1989 3	**Articles:** A Report on Pick's R.83 - Version 3.0 • McDonnell Douglas Release 7.0 • Creating Fault Resilient Pick Applications • Barcode Scanning Equipment • TheCorrelative What, When, Why • Step-by-Step Success for Ultradata Client • Pick Programming Techniques • TCL Talk - Communicating With Others • Useful Functions for Trimming Unwanted Data From Strings		
Sept/Oct 1989 4	**Articles:** The Stratus PICK/VOS Software Architecture • Technology for Technology's Sake • Tree Structured Indexes for the Pick Operating System • Reading Barcodes with PICK Systems • More about the PICK Correlative • How to write a Realistic Benchmark • Toolbox Utilities - An Enhanced SET-FILE Processor		
December 1989 5	**Articles:** Mentor M/x: The Operating System • Data Integrity & REALITY • Minimizing System Degradation • IBM's RISC Storage Architecture • $INCLUDE for the Rest of Us • Code Segments I Have Seen • The Looping Prestore Command • Toolbox Utilities - DOS-like DIR		

The Educational Journal Dedicated to Users of the PICK and PICK/UNIX Computer Systems

TO ORDER:

Limited quantities available of each issue.

Please order by number.

Please send me the following issues @ $6 each (U.S., Canada & Mexico) and $8.50 (Other countries)

Name: _____

Company: _____

Mailing Address: _____

City/State/Zip: _____

Telephone: ( ) _____

SEND TO: **Thurman Marketing Services**
23181 Verdugo Drive, Suite 104A • Laguna Hills CA 92653

PICK is a trademark of Pick Systems, Irvine, CA

Keyboard templates may be assigned by either terminal type or specific port#.

-How do I...?

When in doubt use the HELP function key, normally Ctl-W (ie: What's going on?). HELP is available for simple editing, command line actions, and menus as well. Initially you should use this feature often as it will decrease the time it takes to become fully familiar with ED +.

-Moving Around-

Moving through text is very easy with cursor movement by character, word, line, page, partial page, and context. Additionally, using the GOTO key, you can quickly jump to the top or bottom of the text, start or end of the current line, a specific line (by number), or even a specific horizontal or vertically and horizontally and you"ll find that moving anywhere in the text is very quick and easy.

-Modifying Text-

ED + comes with a full complement of functions for modifying the text of your item. Characters can be inserted or overtyped (using the INSert/REPlace toggle). Characters can be deleted using the Backspace, DELete character, or DELete to End-of-Line keys. Lines may be inserted using the INS LINE or <RETURN> keys, and deleted using the DEL LINE key. <RETURN> can also be used to split a line at the current position with the Backspace or DELete character keys used to join lines.

Pressing Control-T will transpose (switch) the line that the cursor is on with the line following. Combined with the use of the curor keys, this can be used to quickly and easily change the order of the lines of text.

The BLOCK function though, is one of the real powers of ED +. BLOCK provides the ability to mark, copy, cut, paste, move, edit, and store up to 99 different blocks of text during the editing session. A "block" may be a single character, word, line, several lines, or a complete program. Using BLOCK, it's no longer a hassle to move blocks of code around or transfer a routine from one program to another. It's so easy that you'll wonder how you got along without it!

In case that's not enough for you, ED + has a Search and Replace function that allows you to change text locally to a defined block, globally, or selectively as needed - quite a step up from R/old/NEW!

-Macros and Hot Keys-

ED + has a complete MACRO function that will allow recording, playback, editing, storage, and recall of your keystrokes. MACROs also allow the use of variable parameters which will stop the MACRO in progress and request user input. There are 10 HOT KEYs available which can be used to define 10 commonly used functions/MACROS that have not already been assigned to regular ED + function keys. When you combine the MACRO capabilities with the HOT KEY function the possibilities are almost limitless.

-More Nice Stuff-

If you get lost there are the WHERE and STATUS keys to let you know exactly what's going on. You have real-time control of the ENVIRonment for such things as Timed Backup, Case Sensitivity, Search and Replace mode status, Word Delimiters, and one other very nice feature... Smart Indenting, as you enter text. With the INDENT turned on ED + will indent/outdent in the traditional formatted structure automatically, as you type. There's also a Source Code Formatter built in that is accessible from the FILE function menu. The FILE function in itself contains a wealth of options: File, Save, Compile (with Catalog option), Rename, Load, Merge, etc. There's also a TCL CALL function that

allows you to perform any valid Recall command. This feature should be of particular interest to most technicians and special feature programmers. With CALL, if Ed + doesn't have a feature you want, you can write your own routine and link it into ED + a if it were always there (without having to recompile ED + !).

Overall Impressions

In case you haven't guessed, I'm thoroughly impressed with ED +...and I don't impress easily! Just as the world isn't perfect, the same applies to ED + but it's pros decidedly outweigh it's cons. the above synopsis of the many features available are the PROs of ED +, and I've probably overlooked a few. The CONs are mainly in the area of documentation, which although good, could and will be better. Kevin King has assured me that the documentation is being revised for Release 18 and will contain many more detailed explanations and examples that have been requested by users.

Most systems people will immediately be concerned with 2 question:

1) COST - I won't quote pricing but I can assure you that you pay a programmer more in 1 day than ED + will cost you and it's licensed on a per system not a per user basis!

2) SUPPORT - Full support is provided by the author, with regular updates provided to registered systems. ED + is also distributed with full source code at no extra cost to you, just in case. I've already made some changes for my system, due to business requirements, that will be incorporated into Release 18.

If you're skeptical still, how does over 40 installed systems, with several hundred users sound? Or, how about being included in an ED+ Users Network that spans not only the U.S. but the world? If you're truly interested in increasing programmer productivity, you simply must try ED +. I believe you'll be as surprised and pleased as I was. ED + is definitely a "most-favored" addition to my toolbox!

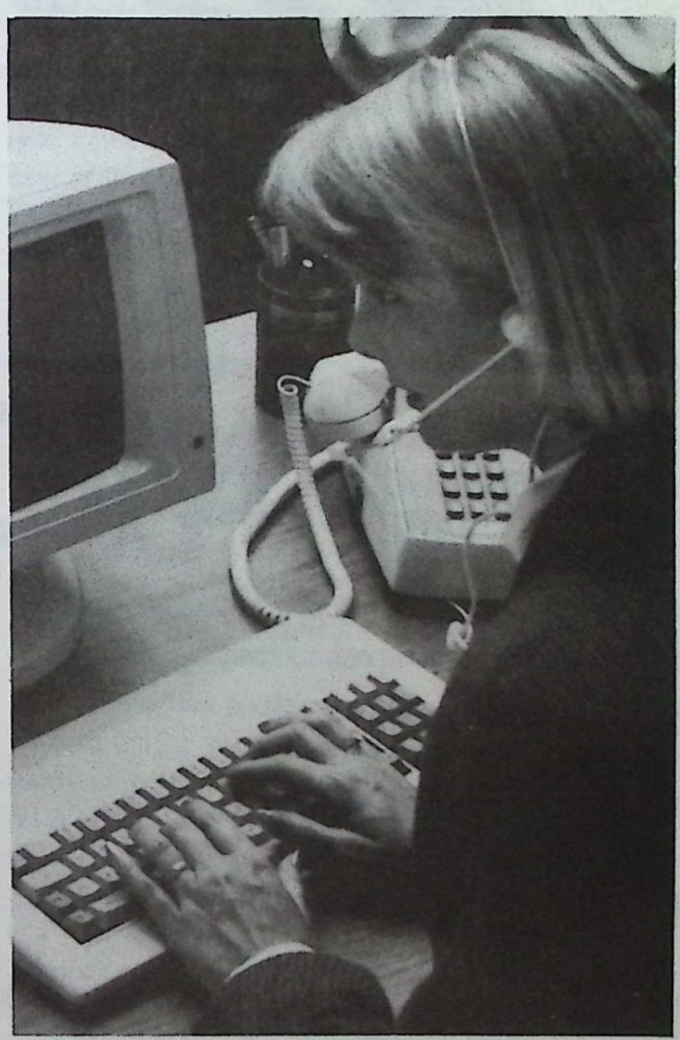
ED + is available directly from: Kevin King
4903 W. 34th Ave.
Denver, CO 80212

Sheldon Markesteyn-Smith has been a programmer and consultant on the Pick (tm) O/S for over 14 years. His background covers the areas of both applications and systems programming, hardware and communications, and he has worked with most PICK O/S implementations.
He is currently employed as a Programming consultant for Xerox Xorp. in Rochester, NY. and can be reached at (716) 427-5613.

The Answer to your High-Volume Telephone Contact Needs.

TELE-BEST

Tele-Best Controls and Improves Contact Process from Selection Thru Follow-Up



Tele-Best is a fast, efficient, effective way to automate your high-volume telephone contact process for maximum productivity and control.

Tele-Best is equally effective with all types of high-volume telephone contact needs, including:

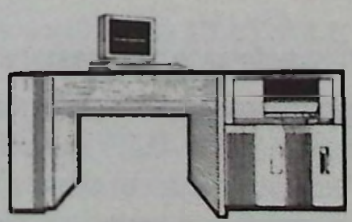
- Account Receivables
- In-House or Agency Collection
- Telemarketing – Sales Prospecting
- Customer Service and Follow-Up
- No Lost Calls
- Complete Contact History
- Customized Environment for *your* PICK Application(s).

Command Kernel-History, Data Base, Word Processor, Auto Dialing

Information and Productivity Systems from

ABEST COMPUTER SYSTEMS INC

513-753-9000



Authorized ADDS Marketer ADDS Applied Digital Data Systems Inc. A Subsidiary of NCR Corporation

Announcing . . .

SINC '90

SYSTEMS INTEGRATION NETWORKING CONFERENCE

IN CONJUNCTION WITH: MICRU International

The international conference and exhibition based on systems integration, data base comparisons, networking solutions and communications options will be held:

Dates: September 17-21, 1990

Where: IRVINE HILTON, Irvine, California

Travel arrangements with AA American Airlines

This conference promises to be the most widely endorsed conference in the history of the Pick community

Booth and exhibition space information can be obtained by calling Bill Thurman at 714/855-4442.

The conference emphasis will be on the integration of current Pick-based systems with applications software and other data base management systems running under other operating systems including UNIX and DOS.

If you are feeling overwhelmed by all the computer alternatives and are looking for practical answers . . . this is the conference you should plan to attend.

Speakers are currently being interviewed by Randall and Kandi Christian at Classic Solutions, 714/854-3664.

Sponsored by: Thurman Marketing Services
News & Review
National Association of Pick and Classic Solutions

File Sizing Faceoff

by Steve Alexander, Triadyne of America
& Brian Gulino, Generation Research

Most PICK system users understand the value of maintaining correctly sized data files. Two software Products available to end users are offered by Brian Gulino, author of File Sizing Tool [FST], and Steve Alexander, author of Autosize [AUTO]. Here they discuss the merits and drawbacks of their products. Which is better? You decide.

Question: In 25 words or less, describe the differences between the two products.

Brian [FST]: The File Sizing Tool takes less knowledge, effort, and time to install and operate. Autosize allows more control of file sizing but requires a more knowledgeable user.

Steve [Auto]: Autosize has several tools for resizing files, accounts, or entire systems. File Sizing Tool is a sledge hammer which can do only one job.

Question: The File Sizing Tool resizes files based on data in the STAT-FILE. Autosize resized files based on actual data on the system. Discuss these two approaches.

Brian [FST]: We rejected dynamic resizing because we had no way of doing it quickly. Our aim is to have an easy-to-use maintenance tool that is regularly used on a system. With regular use, there is no need to resize files, "on the fly", so to speak.

As a consultant, I concede the need for a tool that resizes files on the fly. The Pick tape implementation on the PC for example, has a bug in it where the tape sometimes gets lost in the middle of the file save if it's saving large, severely undersized files. I've been called in to repair this problem twice in the last six months, and it would have been nice to just run a program to resize all undersized files dynamically, especially since the tape save and restore process wasn't working.

Steve [Auto]: In case some of our readers are not familiar with the STAT-FILE, it is the file which is updated with statistics about each file when a FILE-SAVE is done. STAT-FILE contains information like number of bytes in file, number of items, current modulo, and current separation for each file on the system.

With that in mind, it seems to me that there are many possibilities for problems if you rely on the STAT-FILE for accurate information as to what's really on the disk. For example, what if a FILE-SAVE was done then someone ran an update process which changed some files? What if a new account were restored sometime after the last FILE-SAVE? What if Sally forgot to do the FILE-SAVE last Friday' - stranger things have happened.

Since there are so many ways which STAT-FILE could be wrong, I think it's more sensible to resize files as they are, rather than assume that everything is just like it was before the last FILE-SIZE. Relying on the STAT-FILE gives rise to many problems which are easily avoided by using the dynamic

method. I wanted Autosize to solve problems, not create them, so it doesn't use the STAT-FILE at all. As for speed, dynamic resizing is only as fast as your disk. Generally, it beats resizing the entire system by several hours, however.

Question: Autosize may be exported to and run from other accounts on the system. The File Sizing Tool may only be run on other accounts in the automatic mode in conjunction with a file save. Discuss both approaches.

Brian [FST]: Given the large variety of Pick users, running the tool on multiple accounts seemed impractical. We have Pick users in service bureau environments with 50 or 100 customers with each customer having his own account. In this environment, with perhaps 7 or 8 thousand files you really need a file sizing program. It is however, impractical to resize this many accounts individually.

Let's consider other environments. In a single company production environment, you generally have one person in charge of file sizing. She is going to want to resize all files at the same time anyway, so you might as well have the tool on one account.

Next, you have the software house, program development environment, with each programmer on her own account. While I know programmers who might be a little nervous about files being resized without being under their control, I don't know any programmers who view file-sizing as other than a boring chore that they would prefer to avoid.

Finally, I think it's rude to clutter up people's application environment, with verbs and O-pointers to another account. The account gets moved to another system, the programs don't work, nobody knows why, and so on.

Steve [Auto]: The facility to export and use Autosize on other accounts is there for the case where the system administrator wants to allow trusted users to resize their own files or accounts.

I agree with Brian about most programmers thinking of file sizing as a boring task - until they know about Autosize. Once they get Autosize they actually enjoy keeping their files in trim because it isn't boring anymore, they can do it themselves, and they like having the fastest CRT on the desk.

Question: Autosize allows dynamic resizing and The File Sizing Tool resizes files only in conjunction with a save and restore. Discuss the advantages and disadvantages of each approach.

Brian [FST]: In order to resize a file dynamically, you must have exclusive use of the file, that is, no other user may access the file while it is being resized. Autosize's documentation is specific in warning users not to attempt dynamic resizing when anyone else is on the system.

Since Autosize is sold only on the Pick PC implementation, which up to a month or two ago could only accommodate 16 users, insuring that no one else was on the system is a pretty straightforward procedure. You yell across the hall for everybody to log off. The File Sizing Tool has many 50 user plus installations where users are located in remote locations. Insuring no one is accessing the system becomes much more difficult.

We felt that marketing a project that even had the potential to interfere with the users was asking for trouble. The File Sizing Tool has been in the Pick market over 8 years and a large proportion of our customer base is 2nd or 3rd generation DP managers, that is, the successor of the successor of the successor of the person who actually bought the product. We get calls from people working at com-

panies who found the File Sizing Tool account on their system, and want to know what it does - no installation tape, no documentation no invoice. It turns out they bought the thing six years ago. It's a complement to our software that people hook it into their file save procedure and forget about it but it is a little unnerving to get these calls from people who've had the File Sizing Tool installed and operating every time they do a file-save and haven't known it for the past 3 or 4 years.

Steve [Auto]: When I first wrote Autosize, I began by writing a program which worked very much like File Sizing Tool in that it simply resized every file on the system. I quickly discovered that most of the time what I really needed to do was just to resize on file or one account. Resizing the entire system every few days became bothersome to the point that I avoided resizing until the files really got out of hand and the machine began to slow down. Then I would do the resize. FILE-SAVE, and restore.

I talked with others and found that they had similar experiences. Then I wrote the parts of Autosize to resize individual files and accounts. Once they were in place, I found it much more convenient to resize individual files or accounts than to do the whole system at once. As a result, I did the resizing more often, took less time, and kept the machine in better shape.

As for the dangers of dynamic file sizing, Autosize has been on the market for two years. So far, no one has reported even a minor problem caused by anything in Autosize.

Question: Both products use an algorithm to compute a recommended new size for files. Compare the two algorithms.

Brian [FST]: The File Sizing Tool's algorithm recognizes that Pick allocates space for items in whole frame multiples and resizes files to fit the average number of items into the correct number of frames.

The File Sizing Tool starts out with the two figures describing each file which it gets from the STAT-FILE. These figures are the average item size, and the number of items in each file.

First a measurement I call the ideal items per group is computed. The ideal items per group is number of whole items of average item size which can fit into a frame. This is the frame size divided by the average item size and rounded down to the nearest integer. For example, in a file with an average item size of 140 bytes on a Pick PC with 512 byte frames (500 bytes of which can hold data), the average item size is 500/140 or 3.57. 3.57 is rounded down to 3.

Rounding down is critical in the calculation because Pick performs reads and writes on whole items. The .57, which represents the 4th item in the group, would extend over into a second frame, causing an additional disk access to read or write. So you try for only 3 items in your ideal group.

For files with an average item size larger than a frame, the ideal items per group defaults to 1.

Next, we take the ideal items per group and divide it into the number of items in the file. This yields a number close to the recommended modulus of the file. All we have to do to this number to make it the File Sizing Tool;s recommended modulus is:

1. Multiply it by the 1.15 to make it larger than ideal. We do this for two reasons:

a) If the file fluctuates slightly in size over time, its not being resized on every save and restore. The File Sizing Tool will not recommend a file for resizing if its modulus falls in this 15 percent range.

b) Unless item-id's are fixed length sequential numbers, a perfect distribution of items id's into groups is highly unlikely. Adding 15 percent to the modulus helps insure that the file's groups will not contain more than the ideal number of items for maximum speed of access.

2. Use a minimum modulus of 3. We use a minimum modulus of 3 because its so easy to write many items into a modulus 1 file and suffer the consequences of severe under-sizing. In addition, on implementations without item locking such as the PC implementation, items in modulus 3 files are less likely to be group locked.

3. Round up the final figure so that it is not divisible by 2 or 5.

The Autosize algorithm puts at least 3 items in a group regardless of item size. Although this preserves disk space, it's slower when accessing items at random, by item-ID. Random access is the way most interactive programs read and write file items and interactive programs are where performance is most critical. Three item groups take longer to access items when average item size is 175 bytes or larger. Files with 175 byte items should be sized 2 to a group, as the third item brings you into a second frame. Hauling in that second frame slows you down. Moving up to, say, 500 byes items, and with 3 items in each group, you average 2 frame reads to find an item. And since Pick has to rewrite everything after the item on a write, you average two frame writes every time you write an item.

The File Sizing Tool, on the other hand, would size any file with an average item size greater than 250 bytes to have only one item per group. Barring hashing anomalies which work against both algorithms, you have only one item in each group. So you have one frame reads and one frame writes most of the time on items smaller than one frame.

This performance difference is imperceptible when files are small, processed sequentially, and the system is not under load. In this case, the file winds up in memory very quickly, and stays in memory. Since the file takes up less total space using Autosize's algorithm, it may even out perform the File Sizing Tool's algorithm by a time factor of 10-15%. This performance advantage is reversed when the file is much larger than available memory or when only a few items are being read or written to the file. This second case is the more common pattern of usage on working systems.

Steve [Auto]: The assumption underlying Autosize's sizing algorithm is that most Pick-type systems contain pretty much the same kinds of date. Namely:

1 - data files composed of many small, similar items

2 - program files containing reasonably large items

3 - other files (like word processing and spreadsheet) containing a mixture of small and large items.

Autosize optimizes speed and disk usage for two of the file types, and does as well as can be done with the third. By calculating both modulo and separation, Autosize gives the best speed and use of disk possible with data and program files. Although word processing and spread sheet files will always be more or less a mess as far as file sizing goes, Autosize does the best job that can be done.

The math to do this is surprisingly simple. For files with small records (less than 500 bytes) the modulo is just the byte count divided by 500 and made prime. (Autosize always uses prime numbers for modulos). For files with large records (more than 500 bytes), Autosize will average three records per group and will adjust separation accordingly.

The groups will be in overflow, but this cannot be avoided with large records, anyway. The three record average seems to work well, but this can be changed if the user desires.

The File Sizing Tool calculates modulo only. It does not do separation. The minimum modulo is three (even for empty files), and modulos are sometimes not prime numbers. Modulos are calculated at about 15% over the space taken by actual data in the file. This results in many empty groups (wasted disk space). For example, the File Sizing Tool account as delivered on diskette contains 53,465 bytes and consumes 143 frames. Autosize would resize the File Sizing Tool account to use just over 100 frames, thus saving about 40 frames without giving up any speed.

Speaking of speed, Autosize will produce the fastest overall response time possible on any live Pick system, and it won't sacrifice disk space to do it.

Question: Autosize's algorithm allows a separation greater than 1. The File Sizing Tool's does not. Discuss this.

Brian [FST]: Separations greater than one will keep large items in contiguous frames. Since the major factor in disk access times is head movement, and this reduces head movement on contiguous frame access, this will enhance performance.

On a multi-user system under load, it's probable that the disk head will have moved to service another user between subsequent requests for frames. So performance is enhanced when a user is on a lightly loaded system. I fail to see the value in a performance enhancement which enhances performance only when the system is running fast anyway.

A price paid for this enhancement. Larger separations make for smaller moduli. Smaller moduli are more sensitive to changes in number of items on the system. Naturally, they are also more sensitive to variability in item size. Files which have large average item sizes, the files where a separation greater than one is indicated, usually exhibit more variability in size than files with smaller average item sizes.

Steve [Auto]: Pick machines can perform at peak efficiency only when both modulo and separation are calculated and used. These parameters are actually the height and width of the file being described. Like people, some files work better if they are tall and thin, other work better when they are short and fat. Ignoring separation makes all files short and fat.

Since Autosize uses separation and File Sizing Tool doesn't, a machine using Autosize will be more efficient.

Question: Once and for all, does the modulo of a file really have to be a prime number?

Brian [FST]: At various times, various manufacturers, including Pick Systems, have recommended prime number moduli and furnished little programs to find prime numbers.

Selecting a prime number modulas is recommended in order to improve hash distribution of items in a file, e.g. 110 items in a file with a modulo of 11, a prime number, would tend to hash into neat little groups of 10 items each, whereas 100 items in a file with a modulo of 10 would have 2 items in one group, 18 in another group, and so on.

The standard deviation figure in the ISTAT listing is a measure of this variability, this tendency for items to hash evenly. The lower the standard

deviation, the less variability, the more even the hash.

Several years ago, I tested this by resizing all files on an account to prime number moduli and comparing the standard diviations to the non-prime moduli in the original account. I tested about 100 files and I did not detect a difference in the standard deviation. I did however, detect a significant difference when the modulus is divisible by 2 or 5.

Steve[Auto]: Before writing Autosize, I asked Pick Systems if modulos had to be Prime. They said yes, however, no one could seem to explain why. So Autosize makes them prime in deference to Pick Systems. I know of no mathematical reason why a modulo 10 (which is divisible by both 2 and 5, and is not prime) isn't just as good as say, modulo 11 (which is prime). Incidentally, both 2 and 5 are prime numbers.

Question: Discuss safety issues with your product.

Brian [FST]: In automatic mode, the File Sizing Tool will not make any file smaller. In manual mode, you can make a file which is larger than recommended, smaller. If, in manual mode, you try to make a file smaller than recommended, the program beeps at you, displays a complaining error message, and finally lets you undersize the file.

Automatic mode works fine for companies whose files are increasing in size. Companies whose files are shrinking, need to periodically resize files making them smaller. Since oversized files do not present as serious a problem as undersized files, this needs to be done only every six months or so.

The File Sizing Tool can resize your files so they don't fit on your disk drive. This could happen if you had undersized files and were up to 90 percent of your disk capacity. Fortunately, the restore program has a parameter to ignore resizing information and restore the system without resizing files. The File Sizing Tool resizes files to optimize speed of access, at the expense of the disk space.

Steve [Auto]: Autosize is safe if allowed to operate on its own. Since Autosize looks at actual data on the machine and calculates exactly the right modulo and separation. It also allows you to exempt files from resizing, or to make files bigger or smaller (you can change both modulo and separation) than recommended. This is so that you can give personal attention to those 'unusual' files found on some systems.

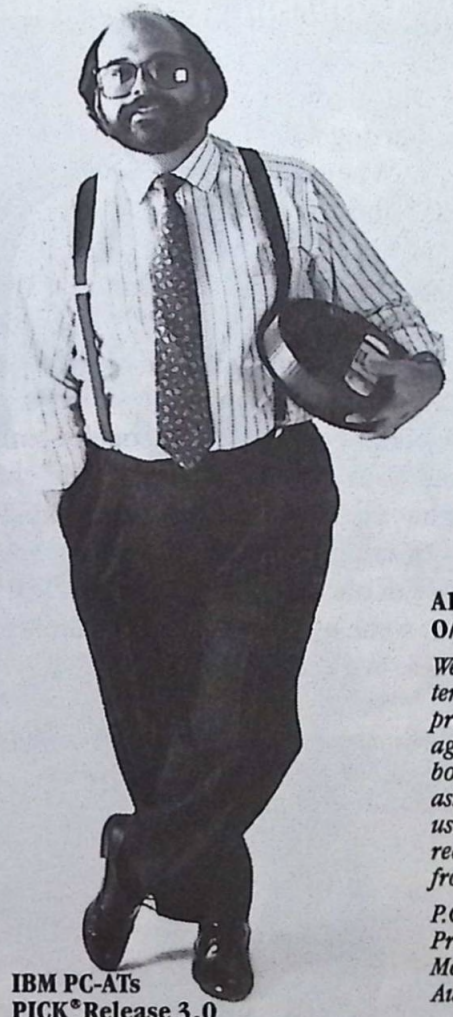
Autosize incorporates no growth factor in its calculations. This means that Autosize will always use exactly the amount of disk that is necessary to hold your data. If you disk is already say 90% full, Autosize will not result in a 'DISK FULL!' message when you try to restore, as File Sizing Tool almost certainly would. In short, Autosize resizes files to optimize both speed and space. Not too big, not too small, just right.

Question: Of what use are the Pick verbs, ISTAT and HASH-TEST, in resizing files?

Brian [FST]: I think they are couterproductive. The existence of the verbs implies two premises that are utterly false: (1) that they should be used, that the system administrator should be carefully HAST-TESTing his files all the time, and (2) that using these verbs will result in a perceptible performance increase. Files are dynamic, and today's HASH-TEST will not work tomorrow. Besides, who has time for this?

We all talk about how Pick is the businessman's operating system which allows us to concentrate on business problems rather than the operating sytem's problems, yet here we have this thing we have to do

ONE WP SOLUTION FITS ALL.



IBM PC-ATs
PICK® Release 3.0

We've been using JET software for almost three years and the current release (The Works 2.0) is by far the most convenient. Its ability to merge mailing lists is particularly useful for our direct marketing activities.

David Lubetkin
Executive Producer
The Image Group
Video/AV production house

ADDS 2500
O/S Release 2.5

We've relied on PICK® based systems for 10 years but found our previous word processing packages cumbersome. So before we bought our current system we asked a number of other PICK® users what WP package they'd recommend. The Works 2.0 from JET was the clear winner.

P.G. "Red" Roberts
President
McLeod Industries, Inc.
Automotive parts manufacturer

McDonnell Douglas 18/600
O/S Release 6.0

We chose The Works 2.0, in part for its DOS-like flavor and function key mode of operation. But also important were its ability to convert and edit our old Ultiword documents and the fact we didn't have to retrain our users.

Marilyn Dupree
Senior Secretary
Shurflo
Pump manufacturer

Honeywell DPS6
with Ultimate O/S

The Works 2.0 was right for us. First, its data base integration capability increases our productivity. And second, its support of laser printers gives us greater control over finished documents.

John Shannon
Manager Information Systems
Microsemi Company
Diode manufacturer

For the solution that fits *your* needs call us at (714) 832-8822 or write:
17632 Irvine Blvd., Ste. T., Tustin, CA 92680, FAX (714) 832-0106.

JET™
SOFTWARE
Word processing for the PICK world.

PC-AT is a trademark of IBM
PICK is a trademark of PICK Systems
Ultiword is a trademark of The Ultimate Corp.

- resizing files - which has nothing to do with the business of the company, it's pure operating system maintenance overhead. The best thing you can say about a product that resizes files is that it does its job without the users knowing it's there.

Steve [Auto]: Though I seldom use HASH-TEST and ISTAT, I guess I'd rather have them and not need them than need them and not have them.

Question: Going toe to toe with competing products like this is unusual in the Pick market. What do you get out of it?

Brian [FST]: A lot. When Steve suggested we do this, I felt a bit superior. I did all this work on the File Sizing Tool and I knew it was better than his Autosize. But after closely examining Autosize, and listening to the reasons he did what he did, I understand that my way is not the only way, and, in many respects not the best way. There are things that Steve did which give me ideas for improving my own product, chiefly in the areas of dynamic resizing, and, as Steve pointed out, in recognizing when you could run out of disk space. Other things he did differently than I which I wouldn't change, but now I'm more aware of the tradeoffs I made.

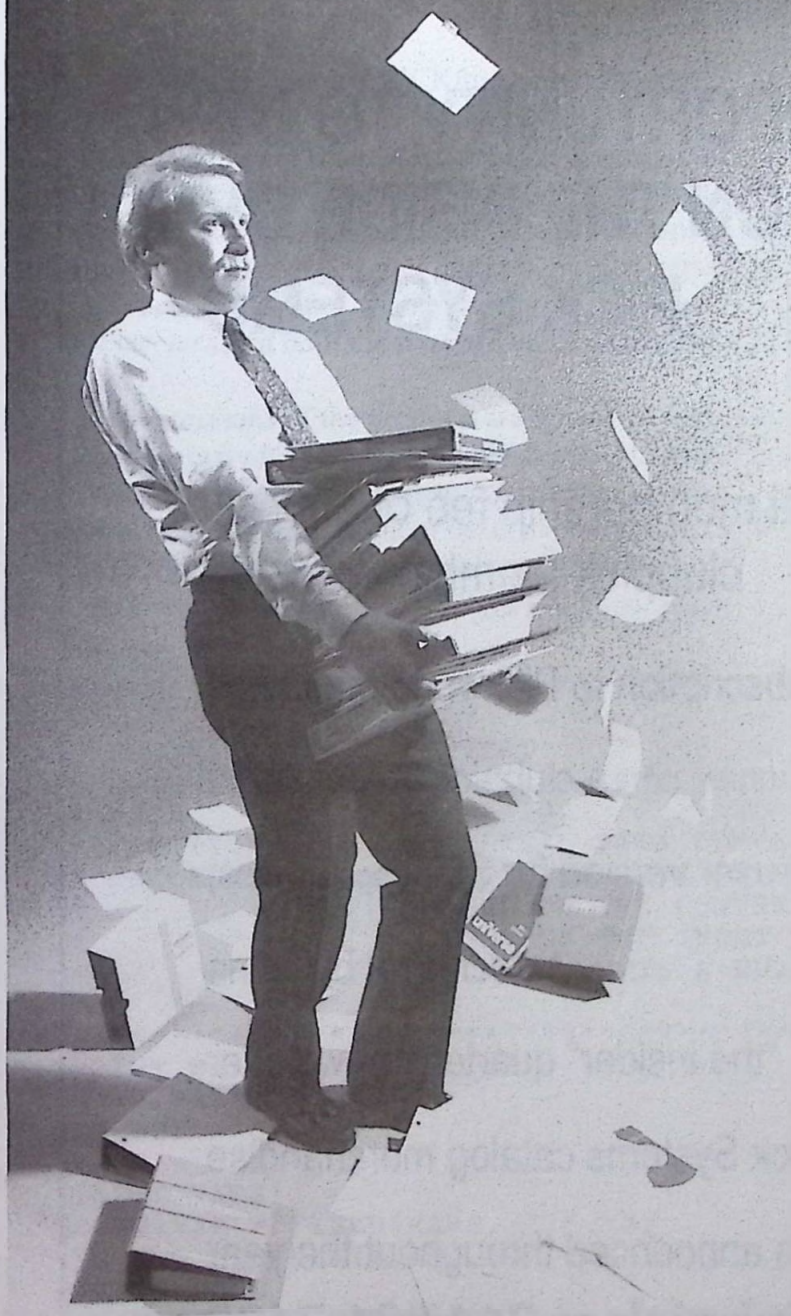
Steve [Auto]: When I was a youngster, I lived on a farm in an area about 30 miles from anywhere. Local people called it 'Plum-nearly', which meant it was plum out of the city and nearly out of the country. We had a dog which, as far as I know, never saw another dog unless somebody with a dog came to visit us. When that would happen, our dog would run round and round in circles, and yelp, and jump, and raise a real ruckus. He seemed very happy to see another of his kind - a kindred spirit.

I think I get that feeling when I meet another person who has worked on something I have. It's always a pleasure to work with someone of similar interest. Since Brian and I both wrote file sizing programs, we have a lot in common - sort of kindred spirits. And of course, I hope this article sparks some interest with our readers in the subject of file sizing, and that some of them purchase Autosize or File Sizing Tool. We sell both products.

For information the authors may be contacted as follows:
Brian Gulino (File Sizing Tool) at (619) 944/6220.
Steve Alexander (Autosize) at (619) 541-1676.

Still Doing It By The Book?

DROP YOUR BOOKS! NOW YOU CAN GET UP TO SPEED ON PICK, JET OR COMPUSHEET+ AT HOME, IN THE OFFICE OR ANYWHERE YOU'VE GOT A PICK MACHINE AND A CRT TERMINAL.



For years we Pick people have struggled to learn the only way we could, by reading the reference manuals. What a lousy way to learn!

Introducing Crescendo's Electronic Tutorials. When you learn by using an electronic tutorial, every terminal is a classroom. And you are learning the way thousands of Pick people have now learned. Each electronic tutorial leads you through simulated, hands-on work sessions in which you actively participate. You try every important command and carefully review the way the system responds to each. It all happens right on the terminal. There is no book to hold in your lap.

Three Crescendo Electronic Tutorials bring you into the world of Pick...

- **The PROF Electronic Tutorial for Pick**
  Learn TCL, EDITOR, ACCESS, PROC even Pick Basic!
  Only $395*
- **The JET Word Processor Electronic Tutorial**
  Also available for UltiWord, DocuMentor, and WordMate.
  Only $295*
- **The CompuSheet+ Electronic Tutorial**
  Many examples and exercises.
  Only $295*

All these tutorials are available for Pick on the XT/AT/386 as well as the Pick-compatible minicomputers. PROF is also available for Prime INFORMATION and VMARK uniVerse.

Get a Free Demo Diskette!

Crescendo Associates wants to rush YOU a demonstration of these exciting products **absolutely free!** The Crescendo Product Presentation Diskette works on any MS-DOS personal computer. You don't even need PICK to run the demonstration.

Just call 1-800-373-1224!!!

So we can rush you your demo diskette and product literature!!!

CRESCENDO Associates, Inc.

*These prices apply to the XT/AT versions. Pick is a trademark of Pick Systems, Irvine, CA.

22000 Springbrook • Suite 205B • Farmington Hills, MI 48024

clubpick®

YOUR OPPORTUNITY TO BE ON THE INSIDE OF PICK SYSTEMS

For an annual membership fee of $50 US, clubpick members receive:

- 1 year subscription to Pickworld magazine
- a clubpick'90 polo shirt
- Advanced Pick single-user version for $50 (regularly $99)
- technical bulletins
- "the insider" quarterly newsletter
- special discounts on Pick Systems catalog merchandise
- plus other benefits announced throughout the year

For membership call clubpick at 714/261-7425

Pick Systems, Inc. • 1691 Browning • Irvine, CA 92714

Conversions for Pick

by Richard J. Sueltenfuss, Consultant

CONVERT is a simple PICK/basic program which can be used to perform a number of ICONV's or OCONV's from TCL. As written, the program can be used to convert dates and times from internal to external, or external to internal formats; to display the hex value of a decimal number, or vice versa; and to perform any of the numberic 'MD' type conversions, in both the internal, and external formats.

Two versions of the program are presented here. The first is for R83, and the second, for OA systems.

The R83 program works in conmucntion with a proc, which accepts input of the conversion type to be executed, and in the case of the 'md' types, an I or O to indicate an ICONV or OCONV. First, place the proc CONV in the MD of your account. Then edit, compile, and catalog CONVERT. To execute the program, enter CONV 'type' at TCL, where 'type' is the conversion to be performed. For the numberic conversions, use MDn,I or MDn,O, where n is the decimal places desired. At the ? prompt, enter the value to be converted, and the result will be displayed on the next line. To end the program, enter a null value (i.e. Carrage Return).

The only difference in the OA version (OA.CONVERT) is that the proc is not necessary. The 'type' parameter(s) can be entered at TCL and the program parses them with the SENTENCE function.

```
CONVERT
*************************************************************************
*
* PROGRAM ID   :  CONVERT
* AUTHOR       :  RICHARD J. SUELTENFUSS
* DATE         :  02/03/90
* FUNCTION     :  UNIVERSAL DATA CONVERTER
* PROCESSING   :  ACCEPTS CONVERSION VARIABLE(S) FROM THE PROC 'CONV', AND
*                  BASED ON THE CONVERSION CODE, PERFORMS AN ICONV OR
*                  OCONV ON THE INPUT VALUE.  CONTINUES THE CONVERSION
*                  PROCESS UNTIL A NULL INPUT IS READ.
*
*************************************************************************
PRINT @(-1)
CONV.VAR = ""
IN.OUT = ""
VARS = ""
INPUT VARS
CONV.VAR = FIELD(VARS, ",", 1)
IN.OUT = FIELD(VARS, "," ,2)
BEGIN CASE
   CASE CONV.VAR = "D2/" ; * 'PICK' DATE TO EXTERNAL FORMAT
      GOSUB 10
```

```
   CASE CONV.VAR = "D" ; * EXTERNAL DATE TO INTERNAL FORMAT
      GOSUB 20
   CASE CONV.VAR = "MTS" ; * INTERNAL, 'PICK' TIME TO EXTERNAL FORMAT
      GOSUB 10
   CASE CONV.VAR = "MT" ; * EXTERNAL TIME TO INTERNAL FORMAT
      GOSUB 20
   CASE CONV.VAR = "MCDX" ; * DECIMAL TO HEX
      GOSUB 10
   CASE CONV.VAR = "MCXD" ; * CONVERT HEX TO DECIMAL
      GOSUB 10
   CASE CONV.VAR[1,2] = "MD" ; * NUMERIC CONVERSIONS
      LOOP
         OK = 1
         BEGIN CASE
            CASE IN.OUT = "O"
               GOSUB 10
            CASE IN.OUT = "I"
               GOSUB 20
            CASE 1
               PRINT @(0,0) : "(O)CONV OR (I)CONV " :
               INPUT IN.OUT
               OK = 0
         END CASE
      UNTIL OK DO
      REPEAT
   CASE 1
      STOP
END CASE
STOP
10 *  CONVERT INTERNAL FORMAT TO EXTERNAL FORMAT
DONE = 0
LOOP
   INVAR = ""
   INPUT INVAR
   IF INVAR = "" THEN
      DONE = 1
   END
UNTIL DONE DO
   OUTVAR = OCONV(INVAR,CONV.VAR)
   PRINT OUTVAR
REPEAT
RETURN
20 *  CONVERT INVAR FROM EXTERNAL TO INTERNAL FORMAT
DONE = 0
LOOP
   INVAR = ""
   INPUT INVAR
   IF INVAR = "" THEN
      DONE = 1
   END
```

```
UNTIL DONE DO
   OUTVAR = ICONV(INVAR,CONV.VAR)
   PRINT OUTVAR
REPEAT
RETURN


OA.CONVERT
*****************************************************************************
*
* PROGRAM ID   :  OC.CONVERT
* AUTHOR       :  RICHARD J. SUELTENFUSS
* DATE         :  02/03/90
* FUNCTION     :  UNIVERSAL DATA CONVERTER FOR OA SYSTEMS
* PROCESSING   :  ACCEPTS CONVERSION VARIABLE(S) FROM TCL, AND,
*                  BASED ON THE CONVERSION CODE, PERFORMS AN ICONV OR
*                  OCONV ON THE INPUT VALUE.  CONTINUES THE CONVERSION
*                  PROCESS UNTIL A NULL INPUT IS READ.
*
*****************************************************************************
PRINT @(-1)
CONV.VAR = ""
IN.OUT = ""
CONV.VAR = SENTENCE(1)
IN.OUT = SENTENCE(2)
BEGIN CASE
   CASE CONV.VAR = "D2/" ; * 'PICK' DATE TO EXTERNAL FORMAT
      GOSUB 10
   CASE CONV.VAR = "D" ; * EXTERNAL DATE TO INTERNAL FORMAT
      GOSUB 20
   CASE CONV.VAR = "MTS" ; * INTERNAL, 'PICK' TIME TO EXTERNAL FORMAT
      GOSUB 10
   CASE CONV.VAR = "MT" ; * EXTERNAL TIME TO INTERNAL FORMAT
      GOSUB 20
   CASE CONV.VAR = "MCDX" ; * DECIMAL TO HEX
      GOSUB 10
   CASE CONV.VAR = "MCXD" ; * CONVERT HEX TO DECIMAL
      GOSUB 10
   CASE CONV.VAR[1,2] = "MD" ; * NUMERIC CONVERSIONS
      LOOP
         OK = 1
         BEGIN CASE
            CASE IN.OUT = "O"
               GOSUB 10
            CASE IN.OUT = "I"
               GOSUB 20
            CASE 1
               PRINT @(0,0) : "(O)CONV OR (I)CONV " :
               INPUT IN.OUT
               OK = 0
         END CASE
```

```
      UNTIL OK DO
      REPEAT
   CASE 1
      STOP
END CASE
STOP
10 *  CONVERT INTERNAL FORMAT TO EXTERNAL FORMAT
DONE = 0
LOOP
   INVAR = ""
   INPUT INVAR
   IF INVAR = "" THEN
      DONE = 1
   END
UNTIL DONE DO
   OUTVAR = OCONV(INVAR,CONV.VAR)
   PRINT OUTVAR
REPEAT
RETURN
20 *  CONVERT INVAR FROM EXTERNAL TO INTERNAL FORMAT
DONE = 0
LOOP
   INVAR = ""
   INPUT INVAR
   IF INVAR = "" THEN
      DONE = 1
   END
UNTIL DONE DO
   OUTVAR = ICONV(INVAR,CONV.VAR)
   PRINT OUTVAR
REPEAT
RETURN
```

```
CONV
PQ
S2
HCONVERT
STON
A"2
P
```

About the author:
Richard J. Sueltenfuss has eight and on half years of data processing experience, the last five and one half of which have been working with PICK on platforms ranging from the IBM PC/XT to the Sequoia Series 300. He is currently an independent consultant working in the Boston area and may be reached by phone at (508) 528-5396 or by writing to:
Mr. Richard J. Sueltenfuss
15 Union Street Norfolk, MA 02056.

TOOLBOX -

Assigning and Controlling Sequential Keys

by Steven Davies-Morris

There are many reasons for assigning sequential keys (Item Identifiers) to records (Items) stored in a Pick data file. Some of these have to do with simple convenience, while others have to do with obtaining the optimum distribution of records throughout a file. This issue's TOOLBOX will not be concerned with the nuances of file-sizing (although there is a good file-sizing program in the TOOLBOX.BP file) because others have covered that very subject in this issue. Instead, the TOOLBOX offers two function subroutines for the control of sequential assigned keys. The subroutine TB.GET.NEXT.ID performs the assignment of the nexzt key by use of either the next sequential number or the next key from a FIFO (First In First Out) stack of "garbage-collected" discarded keys. The subroutine TB.ABANDONED.CNTL performs the "garbage-collection" on any keys which were assigned and then discarded.

The process of designing "good" versus "Bad" keys has been a source of confusion for many otherwise competent Pick programmers. As a general rule of thumb, the Pick machine prefers keys which are right justified numbers filled with leading zeroes. Rather than have a lecture on the exact mechanics of the item-id hashing algorithm, let us reduce the rules for constructing keys to two simple truths:

a) Keys with the same suffix will clump together in a file. So always prefix a key rather than suffix it.

(Example: use -

TB.GET.NEXT.ID, and TB.ABANDONED.CNTL,

intead of -

GET.NEXT.ID.TB, and ABANDONED.CNTL.TB).

This is, of course the exact opposite to the MS-DOS method of appending a file extension to the end of a file name.

b) As the length of the key increases, the significance of the leftmost characters in the string decreases. So keep the length of keys as short as is realistically possible, while making them meaningful.

While these statements are a simplification of what the systems analyst needs to remember when designing keys for a Pick data file, ignoring them is one certain way to shoddy database design. While records might have attributes broken out into other records if necessary, it is rare that a production system can afford the luxury of a redesign because of the poor structure of a key. Those of you who wish to learn more about the fine art of designing keys and sizing files should read PICK FOR PROFESSIONALS: ADVANCED METHODS AND TECHNIQUES by Harvey E. Rodstein. Anything that I might choose to say on the matter has been stated most eloquently in this book.

Let us deal with these two processes in reverse order. Each requires the existence of the other for a closed-loop to be in effect. Without the "garbage-collection" subroutine, the "getnext" subroutine , the FIFO stack updated by the "garbage-collection" subroutine would get larger and larger- just like the Blob, threatening to engulf everything within its reach!

Five arguments are passed into TB.ABANDONED.CTL. The documentation on each variable is contained within the source code to the program. The Processing is straight-forward. In line 65 the primary data file of the application is

tested to see if the discarded key leads to a record which actually exists on file. If it does, then this subroutine was called in error and control is immediately returned to the calling program. To illustrate how this subroutine should be called, let us examine the following hypothetical code segment:

```
LN#   Code
850   IF new.item.flag AND discard.flag THEN
851      CALL TB.ABANDONED.CNTL (po.head-
er.file, control.file,po.header.key, "ABANDONED
*P0*1", "AL")
852   END
```

Lines 69-79 handle the actual control record read logic. Note the proper use of locking logic so that the terminal doesn't sit and beep annoyingly at a user while waiting for a group/item to unlock. The TB.TRIM.A function subroutine called in line 75 is a TOOLBOX subroutine from an earlier issue of the DIGEST which removes all occurances of a specified substring from a source string. Lines 83-85 handle the actual insertion of the abandoned key into the abandoned stack. Lines 89-91 wrapup the subroutine and exit.

TB.GET.NEXT.ID is a more complex piece of code. This is mainly because it must perform two distinct operations. The first involves determining if any abandoned keys exist for it to consume as "NEW" keys. The second only occurs if no abandoned keys are found. In which case the next sequential number from a stored counter will be used as the key, after which the counter is incremented and written back to disk.

This subroutine is passed five arguments, documented in the header block of the source code (to the TOOLBOX release 2.2 standard). Lines 64-72 setup several variables used during processing. Line 73 transfers control to the internal subroutine 1000* in which the control record for this data application's primary file is read into workspace. Note that the locking logic used here follows the same pattern as that of the "garbage-collection" subroutine. The only difference is that this time the user is given the opportunity to quit the entry process if they do not wish to wait for the next available key.

The loop in lines 79-83 consumes each key from the abandoned stack, validating that it is not a legitimate record found in the data file. We do not want to erroneously write over an existing record! Lines 87-93 wrapup the control record and write it back to the disk. Line 99 tests to see if a key has been assigned. If it has then control passes back to the calling program since there is no further need for processing.

If no key has been assigned at this point, then a new control record must be read into workspace. This is the next sequential counter for the data entry application. The loop in lines 110-117 masks the counter to be a right justified leading zero-filled number, concatenates that to the end of the data key prefix - which could be a null if the counter was always just the sequential number - and then makes a read of the data file to see if the record actually exists or not. If it does, then the counter has one added to it, and the process begins again. Lines 121-123 wrapup the control record and exit.

The following hypothetical code segment shows how to call this function:

```
100*  Prompt for keystep
PRINT (xpos,ypos) : prompt.mask : (xpos,ypos) :
INPUT po.header.key :
new.item.flag = false
IF po.header.key = "NEW" THEN ; Get next id
   new.item.flag = true
   CALL TB.GET.NEXT.ID(po.header.file,control.file,
   po.header.key, "PO*1", "R%4") ;*Assign key
    IF INDES(escapes, po.header.key, 1) ;* No NEXT key
assigned
   restart = true
END
IF restart THEN GOTO 100 ;* Reprompt for item key
```

In both function subroutines, the HISTORY: START and HISTORY: END lines mark text which can be extracted for inclusion in a revision control manual. The NOTES: START and NOTES: END lines mark text which can be extracted for inclusion in a function library manual, with variables individually explained. The INCLUDE segments are intended for general use in all TOOLBOX programs. Feel free to use them in your own coding.

Author's footnote"

I would like to thank those of you who took the time to write and order the TOOLBOX, and offer suggestions for new functions, and valid criticisms of minor inefficiencies in the code and bugs in the supporting text. With the next issue of the PICK USER DIGEST I shall include one or two alternatives to TOOLBOX code offered by readers of the column.

I would also like to offer a few words of thanks to Ken Cassady who has gone on to join the staff of Informative Research. He has been a most capable collaborator for the TOOLBOX. I would hazard a guess that the casual observer could not tell the difference between our code! He will be contributing from time to time in the future.

Mr. Davies-Morris is an instructor for JES & Associates and a freelance contracted computer professional. He may be contacted at:

DMCG
3941-B South Bristol, Suite #52
Santa Ana, CA 92704 (714) 665-7286

TOOLBOX source code is available on standard-density floppy diskette for $55.00; half-inch tape (1600 BPI, block-size 8000) is also available for $55.00. It includes a wide range of functions, tools and utilities, built around a powerful shell processor and screen handler. Checks sent as payment for the TOOLBOX should be made payable directly to "Steven Davies-Morris."

```
    TB.GET.NEXT.ID
001 SUBROUTINE TB.GET.NEXT.ID(data.file, control.file, data.item.id, control.item.id, mask)
002 ***
003 * STEVEN DAVIES-MORRIS
004 * A TOOLBOX UTILITY PROGRAM
005 * GETS THE NEXT SEQUENTIAL ITEM ID WHEN THE DATA ITEM IS "NEW"
006 * RELEASE LEVEL: V2.2
007 ***
008 * PROPERTY OF DMCG, 3941-"B" SOUTH BRISTOL, SUITE # 52,
009 *                   SANTA ANA, CA, 92704.
010 *                   (714) 665-7286
011 ***
012 * HISTORY: START
013 * ==============
014 * 12/28/89 - PICK USER DIGEST VERSION PREPARED TO V2.2 STANDARD
015 * HISTORY: END
016 ***
017 * NOTES: START
018 * ============
019 * THIS SUBROUTINE IS A COMPLEX FUNCTION PERFORMING TWO SEPERATE OPERATIONS
020 * TO RETURN THE NEXT AVAILABLE ITEM ID INTO AN APPLICATION. THE FIRST STEP
021 * CHECKS TO SEE IF THERE ARE ANY ABANDONED KEYS WAITING TO BE CONSUMED. IF
022 * THERE ARE, THEN THE FUNCTION RETURNS THE FIRST AVAILABLE ABANDONED ID.
023 * IF NOT, THEN THE SECOND STEP IS PERFORMED. THE SECOND STEP RETREIVES THE
024 * NEXT SEQUENTIAL ITEM ID COUNTER ASSOCIATED WITH THE APPLICATION, AND
025 * READS A DUMMY RECORD FROM THE DATA FILE UNTIL IT CANOT READ A RECORD. AT
```

```
026 * THIS POINT IT RETURNS THE CREATED ITEM ID.
027 * ARGUMENTS PASSED INTO THIS SUBROUTINE ARE:
028 * 'data.file'       - MAIN DATA FILE ASSOCIATED WITH THE CALLING PROGRAM.
029 *                     BEFORE UPDATING THE PROPER ABANDONED KEY STACK, THIS
030 *                     FILE IS CHECKED FOR THE EXISTENCE OF A RECORD WITH
031 *                     THE KEY THAT IS BEING DISCARDED.
032 * 'control.file'    - THIS FILE CONTAINS THE CONTROL RECORD HOLDING ANY
033 *                     KEYS ASSOCIATED WITH THE MAIN DATA FILE WHICH HAVE
034 *                     BEEN ABANDONED.
035 * 'data.item.id'    - KEY WHICH WILL BE VALIDATED AGAINST THE data.file.
036 *                     IF NO RECORD EXISTS IN THE DATA FILE WHICH MATCHES
037 *                     THIS KEY, THEN THE KEY WILL BE INSERTED INTO THE
038 *                     ABANDONED STACK CONTROL RECORD (control.item). THIS
039 *                     WILL CONTAIN THE PREFIX OF THE KEY. THIS SUBROUTINE
040 *                     WILL TAKE THE NEXT SEQUENTIAL COUNTER AND BUILD A
041 *                     NEW DATA KEY BY CONCATENATING IT AND THE PREFIX.
042 *                     FORMAT: PREFIX : COUNTER (EG: "PO*1*" : 0298).
043 * 'control.item.id' - KEY TO THE CONTROL RECORD WHICH CONTAINS data.file
044 *                     KEYS WHICH HAVE BEEN ABANDONED. THIS WILL CONTAIN
045 *                     ONLY THE SUFFIX OF THE KEY. THIS SUBROUTINE WILL
046 *                     TAKE THE WORDS "ABANDONED" AND "NEXT" AND BUILD A
047 *                     NEW KEY BY CONCATENATING THE SUFFIX TO THE WORD.
048 *                     FORMAT: WORD : SUFFIX (EG: "ABANDONED" : "*PO*1").
049 * 'mask'            USED TO FORMAT THE SEQUENTIAL ITEM COUNTER SO THAT
050 *                     DISTRIBUTION OF ITEMS IN data.file ALLOWS FOR THE
051 *                     HASHING ALGORITHM'S RIGHT-HAND WEIGHTING. FORMAT IS
052 *                     "R%n", WHERE R = RIGHT JUSTIFIED, % = ZERO FILL, AND
053 *                     n = THE NUMBER OF CHARACTERS THE COUNTER TAKES UP.
054 * NOTES: END
055 ***
056 * STANDARD INCLUDES
057 ***
058 INCLUDE TOOLBOX.INCLUDES EQU.CONSTANTS
059 INCLUDE TOOLBOX.INCLUDES SET.CRT.CLEARS
060 ***
061 * OTHER SETUP
062 ***
063 data.prefix = data.item.id
064 control.suffix = control.item.id
065 bottom.screen = @(0, 23) : clear.line
066 ***
067 * GET THE ABANDONED STACK FOR THIS COMPANY
068 ***
069 got.key = false
070 control.item.id = ('ABANDONED' : control.suffix)
071 default = null ;* If item is not on file
072 GOSUB 1000 ;* Get the control record
073 IF (mode # null) THEN RETURN ;* Get out of Dodge...
074 ***
075 * DO ANY ABANDONED KEYS EXIST FOR USE?
076 ***
```

```
077 IF (control.item # null) THEN
078    LOOP
079       temp.id = control.item(1, 1) ;* Top element from stack
080       control.item = DELETE(control.item, 1) ;* Pop the stack
081       READ temp.item FROM data.file, temp.id ELSE got.key = true
082    UNTIL ((control.item = null AND temp = null) OR got.key) DO REPEAT
083 ***
084 * UPDATE ABANDONED STACK
085 ***
086    IF got.key THEN
087       data.item.id = temp.id
088    END ELSE
089       control.item = null
090    END
091    WRITE control.item ON control.file, control.item.id
092 END
093 RELEASE control.file, control.item.id
094 ***
095 * IF got.key, THEN EXIT TO CALLING PROGRAM, OTHERWISE GET THE PROPER
096 * NEXT SEQUENTIAL COUNTER RECORD AND ASSIGN THE ITEM ID.
097 ***
098 IF got.key THEN RETURN ;* Get out of Denver...
099 ***
100 * SINCE WE GOT TO THIS POINT, DO THE SEQUENTIAL COUNTER STUFF
101 ***
102 control.item.id = ("NEXT" : control.suffix)
103 default = 1 ;* If item is not on file
104 GOSUB 1000 ;* Get control record
105 IF (mode # null) THEN RETURN ;* Get out of Dallas...
106 ***
107 * NOW SEE IF THE DATA ITEM EXISTS
108 ***
109 LOOP
110    data.item.id = (data.prefix : (counter mask))
111    READ temp.item FROM data.file, data.item.id THEN ;* Found
112       counter = (counter + 1)
113    END ELSE
114       got.key = true
115    END
116 UNTIL got.key DO REPEAT
117 ***
118 * UPDATE NEXT SEQUENTIAL COUNTER
119 ***
120 WRITE (counter + 1) ON control.file, control.item.id
121 RELEASE control.file, control.item.id
122 RETURN
123 ***
124 * GET THE CONTROL RECORD
125 ***
```

```
126 1000*
127 LOOP
128    lock.flag = false
129    mode = null
130    READU control.item FROM control.file, control.item.id LOCKED
131       PRINT bottom.screen : 'Please wait for the next.id to become available. Esc to quit: ' :
132       INPUT mode, 1 :
133       lock.flag = NOT(INDEX(escapes, mode, 1))
134       IF lock.flag ELSE data.item.id = mode ;* Tested in main program
135    END THEN ;* We have the record: it is now LOCKED
136       control.item = control.item(1)
137       CALL TB.TRIM.A(control.item, blank) ;* Trim all spaces
138    END ELSE ;* Record didn't exist
139       control.item = default
140    END
141 WHILE lock.flag DO REPEAT
142 RETURN
143 ***
144 * END OF TB.GET.NEXT.ID SUBROUTINE
145 ***
146 END
```

```
    TB.ABANDONED.CNTL
001 SUBROUTINE TB.ABANDONED.CNTL(data.file, control.file, data.item.id, control.item.id, locate.code)
002 ***
003 * STEVEN DAVIES-MORRIS
004 * A TOOLBOX UTILITY PROGRAM
005 * SAVES ITEM IDS WHEN A 'NEW' RECORD IS 'EXITED' RATHER THAN 'FILED'
006 * RELEASE LEVEL: V2.2
007 ***
008 * PROPERTY OF DMCG, 3941-'B' SOUTH BRISTOL, SUITE # 52,
009 *                   SANTA ANA, CA, 92704.
010 *                   (714) 665-7286
011 ***
012 * HISTORY: START
013 * ==============
014 * 12/28/89 - PICK USER DIGEST VERSION PREPARED TO V2.2 STANDARD
015 * HISTORY: END
016 ***
017 * NOTES: START
018 * ============
019 * THIS SUBROUTINE SHOULD BE TREATED AS A FUNCTION. IT PERFORMS ONE
020 * OPERATION - THE MAINTENANCE OF A CONTROL FILE RECORD. WHEN A USER OF
021 * DATA ENTRY PROCESS WHICH HAS ALREADY ASSIGNED AN AUTOMATIC ITEM ID
022 * ELECTS TO DISCARD THE RECORD BEFORE FILING IT, THE ASSIGNED KEY SHOULD
023 * NOT BE LOST. THIS SUBROUTINE WILL KEEP THESE ABANDONED KEYS, ALLOWING
024 * THEM TO BE RECYCLED VIA THE TB.GET.NEXT.ID FUNCTION SUBROUTINE.
025 * ARGUMENTS PASSED INTO THIS SUBROUTINE ARE:
026 * 'data.file'       - MAIN DATA FILE ASSOCIATED WITH THE CALLING PROGRAM.
```

```
027 *                    BEFORE UPDATING THE PROPER ABANDONED KEY STACK, THIS
028 *                    FILE IS CHECKED FOR THE EXISTENCE OF A RECORD WITH
029 *                    THE KEY THAT IS BEING DISCARDED.
030 * 'control.file'    - THIS FILE CONTAINS THE CONTROL RECORD HOLDING ANY
031 *                    KEYS ASSOCIATED WITH THE MAIN DATA FILE WHICH HAVE
032 *                    BEEN ABANDONED.
033 * 'data.item.id'    - KEY WHICH WILL BE VALIDATED AGAINST THE data.file.
034 *                    IF NO RECORD EXISTS IN THE DATA FILE WHICH MATCHES
035 *                    THIS KEY, THEN THE KEY WILL BE INSERTED INTO THE
036 *                    ABANDONED STACK CONTROL RECORD (control.item).
037 * 'control.item.id' - KEY TO THE CONTROL RECORD WHICH CONTAINS data.file
038 *                    KEYS WHICH HAVE BEEN ABANDONED.
039 * 'locate.code'     - USED TO DETERMINE THE ORDER OF INSERTION INTO THE
040 *                   - control.item CONTROL RECORD.
041 * NOTES: END
042 ***
043 * STANDARD INCLUDES
044 ***
045 INCLUDE TOOLBOX.INCLUDES EQU.CONSTANTS
046 INCLUDE TOOLBOX.INCLUDES SET.CRT.CLEARS
047 ***
048 * OTHER SETUP
049 ***
050 bottom.screen = @(0, 23) : clear.line
051 ***
052 * DATA FILE ITEM SHOULD NOT EXIST, BUT TEST FOR IT.
053 * IF FOUND, EXIT IMMEDIATELY TO CALLING PROGRAM.
054 ***
055 READ data.item FROM data.file, data.item.id THEN RETURN
056 ***
057 * GET THE ABANDONED KEY STACK FOR THIS FILE
058 ***
059 LOOP
060    lock.flag = false
061    READU control.item FROM control.file, control.item.id LOCKED
062       PRINT bottom.screen : "Please wait for the next id to become available..." :
063       RQM; RQM ;* Give up the timeslice
064    END THEN ;* We got the record; it is now LOCKED
065       CALL TB.TRIM.A(control.item, blank) ;* Trim all spaces
066    END ELSE ;* No abandoned keys
067       control.item = null
068    END
069 WHILE lock.flag DO REPEAT
070 ***
071 * FIND THE PROPER POSITION FOR INSERTION
072 ***
073 LOCATE(data.item.id, control.item, 1; pos; locate.code) ELSE
074    control.item = INSERT(control.item, 1, pos; data.item.id)
075 END
076 ***
077 * UPDATE ABANDONED STACK
```

```
078 ***
079 WRITE control.item ON control.file, control.item.id
080 RELEASE control.file, control.item.id
081 RETURN
082 ***
083 * END OF TB.ABANDONED.CNTL
084 ***
085 END
```

```
    EQU.CONSTANTS
001 * INCLUDE SEGMENT
002 * SET STANDARD CONSTANTS.
003 EQU null TO ''
004 EQU blank TO ' '
005 EQU true TO 1
006 EQU false TO 0
007 EQU bell TO CHAR(7)
008 EQU form.feed TO CHAR(12)
009 escapes = (CHAR(27) : CHAR(251))
```

```
    SET.CRT.CLEARS
001 * INCLUDE SEGMENT
002 * SET CRT CLEARS STUFF
003 clear.screen = @(-1)
004 clear.down = @(-3)
005 clear.line = @(-4)
```

Subscribe today!

When it's happening in the PICK or PICK/UNIX Environment . . .

You'll read about it in:

NEWS & REVIEW

THE MONTHLY NEWSPAPER for the PICK and PICK/UNIX COMPUTER SYSTEMS

NCE Introduces Hardware Support for the 90's

Free! (US only)

Randy Wolfgram, an NCE Depot product specialist, troubleshoots a CPU board.

AND . . .

Yes! Please enter my subscription(s) as follows:

☐ **News & REVIEW:** Published Monthly

__ First Class Mail - $15/United States Residents

__ Third Class Mail - **FREE** / United States Only

Outside United States:

Payable only by credit cards, money orders or checks drawn on U.S. banks.

__ First Class Mail - $15/U.S. (Canada or Mexico)

__ Air Mail - $20 (United Kingdom only)

__ Air Mail - $30 (All other countries)

I am: (Please check) __ End User __ System Dealer

__ Software Dealer __ PICK Licensee

HARDWARE TYPE: _____

SEND TO:

Thurman Marketing Services
23181 Verdugo Drive •Suite 104A
Laguna Hills CA 92653 • FAX 714/380-3942

☐ **PICK USER DIGEST:** Published bi-monthly

__ $40/U.S. (U.S. & Canada)
Company discounts*: 2-4 subscriptions $35/U.S.each

__ $60/U.S. (All other countries)
Company discounts*: 2-4 subscriptions $55/U.S.each

Name: _____

Title: _____

Company: _____

Mailing Address:

City/State/Zip: _____

Telephone: (___) _____

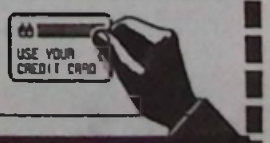
METHOD of PAYMENT: ___Payment Enclosed

-or- Charge to my : ___Visa ___MasterCard ___AMEX

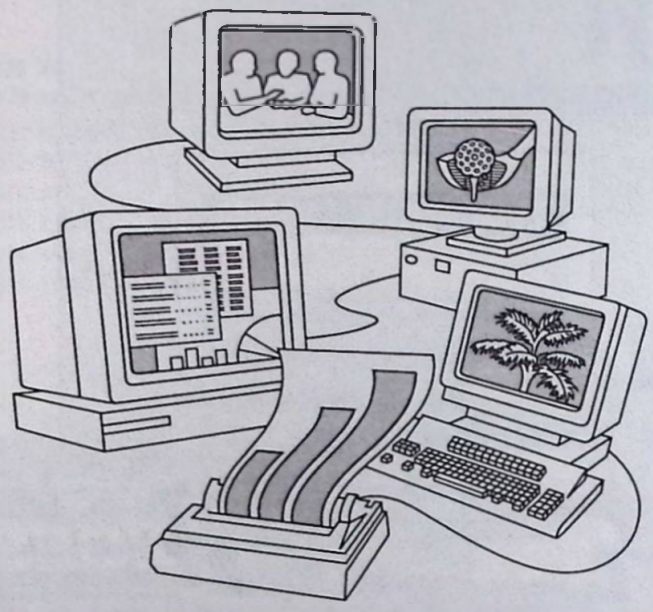
Account #: _____

Expiration Date: ____/____

Name(s) on Card: _____



PICK is a trademark of Pick Systems, Irvine.



Announcing....

The National Revelation Conference

October 29 - November 1, 1990
The PGA Sheraton Resort
Palm Beach Gardens, Florida

Maybe you've heard about it, read about it or even attended last year's event! Well, we're back again this year -- bigger and better than ever. Designed for managers, executives and DP/MIS professionals, the **Second Annual National Revelation Conference** will explore Advanced Revelation and the new technologies available in Release 2.0. As the complete application development environment, you'll see why Advanced Revelation is the optimal environment to develop robust information systems. Don't miss this opportunity to learn, educate and share your ideas with other industry professionals!

Emerging Technologies

- ❑ Environmental Bonding
- ❑ SQL/Client Servers
- ❑ Local Area Networks
- ❑ Bar Coding/OCR
- ❑ New 2.0 Features

Expanded Program

- ❑ Business & technical Sessions
- ❑ Exposition/Demonstration Area
- ❑ Mission Critical Applications
- ❑ Intensive Hands-on Training
- ❑ Special Events, Tennis, Golf & Sailing

Early Bird Registration

	Early Bird	After 6/15/90
Conference 2 days of sessions, demo/expositions	$450	$600
Training 1 full day of intensive hands-on training	$225	$300
Package Complete conference program	$600	$750

Register by June 15 to enter a drawing for a free golf, tennis or sailing prize of *your choice!*

Call Sandy Schiller at **201-795-4003** to register and receive a Speaker Kit!

The National Revelation Conference is co-sponsored by Cogent Information Systems, Inc. and Revelation Technologies, Inc.

Savings on a Silver Platter.

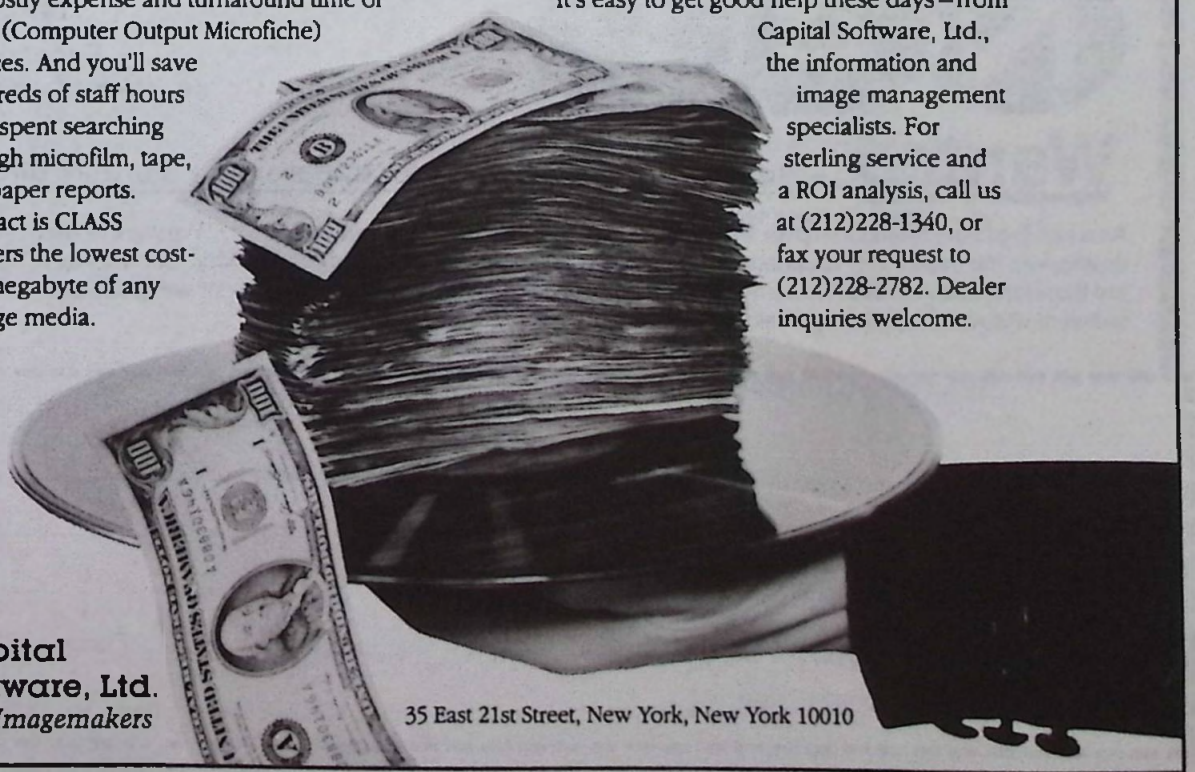
When you've got CLASS,™ you've got savings– handed to you on an optical disk.

CLASS is *Capital Laserdisc Archival Storage System*, the advanced optical technology that gives your PICK system unlimited on-line storage for your data and reports.

With CLASS, you'll save billions of bytes of magnetic disk space – considering that one optical disk can store over 15 million invoices. You'll save thousands of dollars by eliminating the costly expense and turnaround time of COM (Computer Output Microfiche) services. And you'll save hundreds of staff hours once spent searching through microfilm, tape, and paper reports. The fact is CLASS delivers the lowest cost-per-megabyte of any storage media.

The CLASS server caters to multiple users permitting shared access to all data. CLASS comes with a sophisticated toolkit for seamless integration into your existing PICK applications, and it can grow with you, from one individual optical drive to a jukebox to incorporating the latest in scanned image technology.

It's easy to get good help these days – from Capital Software, Ltd., the information and image management specialists. For sterling service and a ROI analysis, call us at (212)228-1340, or fax your request to (212)228-2782. Dealer inquiries welcome.



™ Capital Software, Ltd.
The Imagemakers

35 East 21st Street, New York, New York 10010

Advertiser Index

Attention PICK Users

An online special interest group (SIG) for PICK users is now available on the ProStar Plus network (Seattle, Wa).

ProStar Plus is a regional commercial computer network. Their current rates are $.36 per hour, excluding any telephone charges.

ProStar offers a free introductory package of 2 hours worth of access credit.

Access ProStar at (206) 941-0317 (modem) with communications settings of 300/1200/2400 baud, 8 bits, no parity, 1 stop bit. Leave your PICK questions for other users to answer. Users of any PICK implementation are invited.

The SIG operator is Cary Thomas. He may be contacted at (206) 767-4827 (voice), 6-9 pm, Pacific Time, for further information. Mr. Thomas will not be able to reutrn long distance calls bub may also be contacted online at:Compuserve: 72431,1351; InterNet: 72431.1351@compuserve.com; Prostar: Cary; or PC-Link: CaryCWT

The National Association of Pick

1147 Jackson Place

Escondido CA 92026

619-739-0953 — FAX 619-265-7504

Wanted — Software Directory listings for NAP's on-line Software Directory

As a service to our membership we are providing access to an on-line software directory. We want to afford software developers the opportunity to update or insure that their product is included in our directory. Software developers are therefore invited to submit information to be included in the directory. Complete the following and attach the software writeup. Mail to the address shown above.

Product Name: _____ Company Name _____

Product Function:* _____ Address _____

Target Market: _____ City _____

Contact Person: _____ St _____ Zip _____ Phone _____

* i.e. Payroll, Accounting, Inventory Control, etc.

The word on the street about UNIX is Unisys.

The UNIX® computer operating system lets customers design optimal solutions free of technical constraints, without having to change the way they do business.

Unisys committed early on to providing the best and most comprehensive UNIX solutions on the market, so our customers could create solutions that work.

> *"Unisys has quietly become a formidable player in the UNIX market. They can now address anybody's requirements at a competitive price."*
>
> —Aberdeen Group, as quoted in Computerworld

> *"Unisys expands its UNIX-based U Series, encompassing a desktop-to-whole-enterprise strategy."*
>
> —Information Week

One way we're achieving that goal is by partnering with VMark Software, Inc. and UniData, Inc. to provide PICK-based resellers with tools that support their solutions on Unisys UNIX O/S platforms. Just look at the interest we've generated...

> *"Unisys' commitment to open systems in general, and UNIX in particular, is stronger than ever... those who've seen the company's newest systems...are among the company's most ardent supporters."*
>
> —Information Week

With a focus on complete UNIX solutions like ours, it's no wonder Unisys is the world's number one vendor of commercial UNIX systems. No other company can match our product breadth and software depth, and we are doing it now. Not talking about it for the future.

Get the word for yourself. Contact us today.

PICK-based VARs selling Unisys UNIX O/S solutions include:

- AFTEC, Inc.
- Advantage Information, Inc.
- BIS Sales
- The Coral Group
- GRMS, Inc.
- The Software Group

UNISYS AND YOU.
The power of ²

See us at Spectrum booth #112.

UNISYS

BULK RATE
U.S. POSTAGE
PAID
PERMIT #30391
LOS ANGELES, CA

NEWS & REVIEW
A Publication
Thurman Marketing Services, Inc.
23181 Verdugo #104A • Laguna Hills • CA 92653