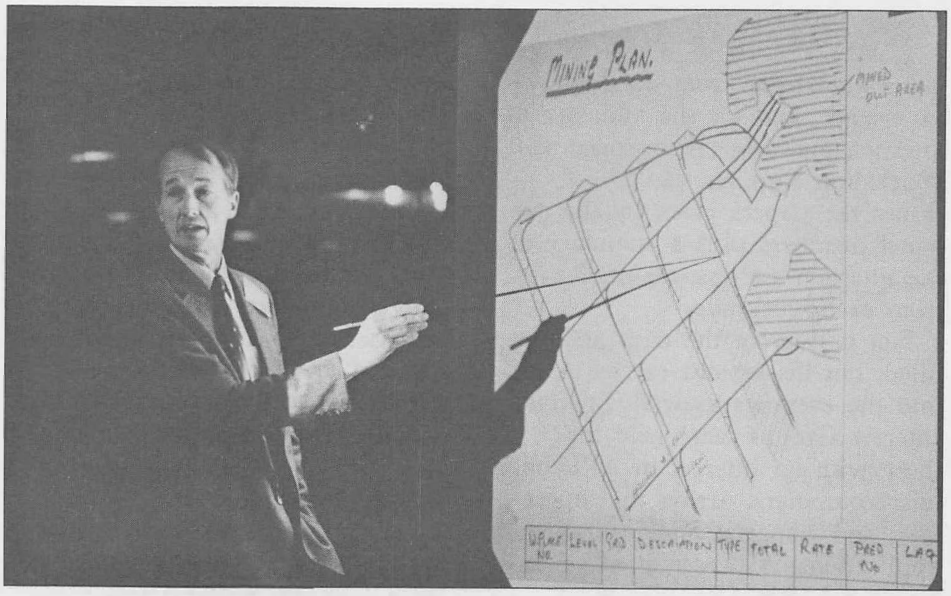
Volume 10/Number 6

November/December 1982

I. P. Sharp Associates NEWSLETTER

In This Issue



1982 APL Users Meeting

Over 700 persons from 22 countries gathered in Toronto for the 1982 APL Users Meeting in October. Lively exchanges among attendees and speakers on the problems, solutions, and applications of APL highlighted this third international users' meeting.

The use of APL is increasing. **Ray Jordan**, editor of *APL Market News*, in his talk estimated there are over 100 000 persons who know some APL. Also the number of APL vendors has grown, from 10 in the late '60s and early '70s, to 140 in 1982. The market for APL is maturing.

From the opening session to the close of the meeting, users asked for more information. And the meeting aimed to do just that. The three-day meeting was preceded by day-long tutorials to introduce APL to actuaries, financial professionals, managers, statisticians and economists. From these tutorials, the following publications are available:

An Introduction to APL for Managers *($5.00)*
An Introduction to APL for Statisticians and Economists *($10.00)*

The most popular sessions were designing APL systems that are user friendly, maintainable, and efficient. These talks organized key concepts in the design of APL systems. Starting with a fundamental design and implementation philosophy, they go on to give you specific recommendations on how to design APL systems that require less maintenance effort and improve program productivity. These Special Technical Topics are compiled in **Volume II of the Proceedings** *($10.00)*.

The workshops on introducing APL, managing APL, and APL training provided an opportunity for panelists to share their cumu-

Something for everyone

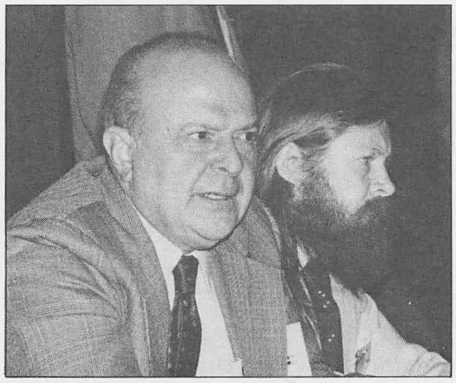
lative knowledge from hands-on experience, and for the audience to question and voice their concerns. **Workshop Reports** *($5.00)* includes the papers given by the panel members plus a summary of the question-and-answer session from each workshop.

Not only were the days action-filled, but discussions carried on into the evenings as well. Special Interest Groups were held for those with an interest in APL on microcomputers, arrays and operators, and actuarial and insurance applications. A presentation on in-house SHARP APL was given and a MABRA users group got together to exchange ideas and applications.

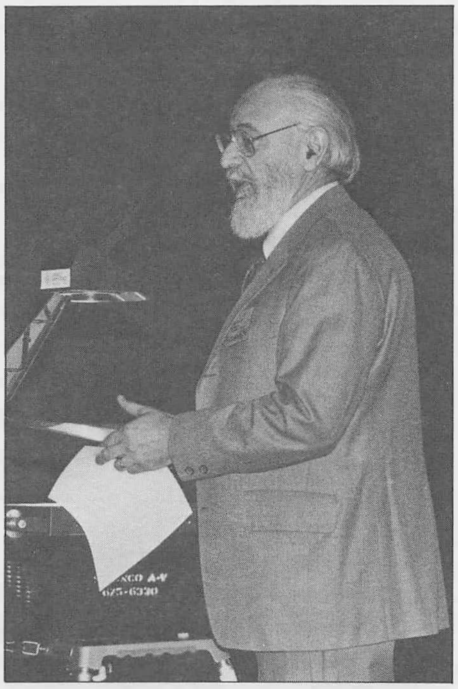
The first night of the conference was highlighted by a special presentation on APL animation in *TRON*. Equipped with slides and film, **Judson Rosebush**, president of Digital Effects Inc., New York, discussed how APL was used to animate sequences for the film *TRON*, as well for logo and scientific animations, and advertisements.

John Corrie *(front page)*, project manager, Mining and Production Services, INFOGOLD, travelled from Welkom, South Africa to participate in the workshop on managing APL. He also presented a paper "The PROMIS Mine Planning and Control System". At INFOGOLD, "We foresee an increasing role for APL because of its well known advantages: fast development, ease of maintenance, ease of change, and shorter staff training."

If you missed the meeting, order your copy of the **Proceedings**. **Volume I** *($15.00)* contains the application papers plus the opening and closing session papers. Copies of all 1982 APL Users Meeting publications are available from the Publications Department in Toronto.



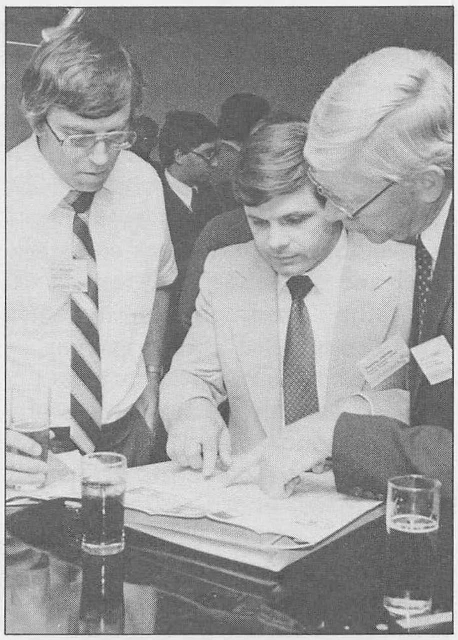
Left: **Roger Moore** *(background)*, I.P. Sharp Associates, Toronto, Ont. and **Michael Montalbano** *(foreground)*, IBM, San Jose, Ca. *"APL provides the best set of gadget-understanding and gadget-controlling ideas currently available. Looking to the future, it provides the best base for the methodology we must develop if we are ever to bring our gadget-based technology under control."*



Above: **Adin Falkoff**, IBM, Yorktown Heights, N.Y. *"Introducing APL is like selling anything. Identify a potential need, do the legwork to find receivers, and make sure that APL is available."*

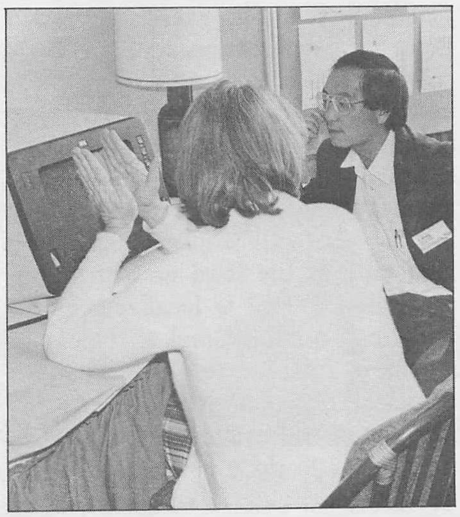
Above: **Eugene McDonnell** *(right)*, I.P. Sharp Associates, Palo Alto, Ca. *"It's possible to make a purely functional language out of APL by adhering to these restrictions: don't use assignment, use the direct definition form of function definition, and don't use global variables."*

And not all work

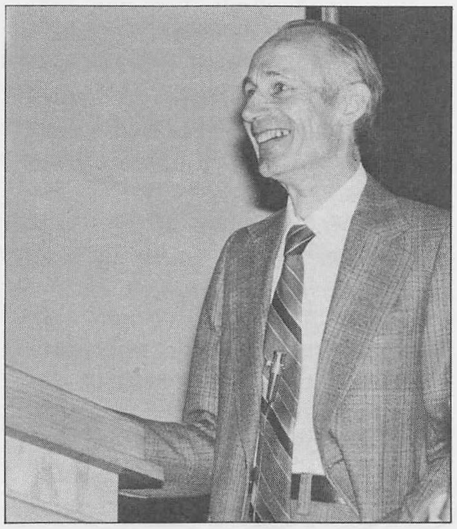


Above: "Best part of the conference is meeting people and talking with people, finding out how they are using APL, and of course, . . . finding a good restaurant."

Below: The SHARP APL graphics exhibit introduced many people to SUPERPLOT.



Below: **Donald McIntyre**, Pomona College, Claremont, Ca. *"APL is not the product of Iverson alone, or of Falkoff and Iverson, or indeed, of any group of living people. It is the result of brilliant insight, careful thought and hard work through at least 5000 years. Ken Iverson is merely the latest figure in a succession that includes Peano, Boole, Sylvester, Cayley, Newton, Leibnitz, Napier, Stevinus, Fibonacci, Diophantus, and the unknown Egyptian, whose work was copied by Ahmes, the scribe. We are indeed in distinguished company. Thank you Ken and Adin for your great contribution."*



APL 82

APL Fans Flock to Heidelberg

Sandra Eadie, Düsseldorf

To the surprise and delight of the organizing committee, 617 persons from 23 countries attended APL 82 in Heidelberg, West Germany, last July.

Conference attendees heard many informative papers, ranging from Philip Chastney's tongue-in-cheek proposal for a new APL entity,. a *snark*, which is a new kind of nothingness, to proposals to index infinite arrays. Enclosed arrays were also discussed in great depth; however, the consensus seems to be to wait and see what the users do with them before any major new changes are announced or standards agreed on.

Although people often complain about the "strange" symbols used in APL, APL's compactness and freedom from the English language are seen as an advantage in non-English-speaking countries. The blind also consider this an advantage, according to Waltraud Schweikhardt, who is blind. By using an extended version of Braille, the programmer can read and write programs much more quickly than in English-based languages.

Copies of the APL 82 Conference Proceedings (ACM Order No. 554820) are available from the Association for Computing Machinery. The price for ACM members is $22.00 U.S.; for non-members $29.00 U.S. Place orders prepaid with:

ACM Order Department
P.O. Box 64145
Baltimore, Maryland 21264

New Release of SHARP APL

Leslie Goldsmith, Toronto

Every six months, I.P. Sharp Associates releases an updated version of SHARP APL to its own timesharing system and to customers who run SHARP APL on their own computers. A new release was installed on the SHARP APL Service in September. In November, this release was distributed to in-house customers.

A new release represents six months of development work on the SHARP APL product. Typically, the release may contain extensions to the SHARP APL language, improvements to the performance of existing features, and remedies for outstanding problems or bugs. The APL product encompasses more than just the language, so areas of enhancement extend to APL software which forms part of the system, to auxiliary processors which provide ancillary services to users, and to batch programs which are required to operate and maintain an APL system but are seldom visible to users.

APL runs faster and in less space using "references"

Significant modifications were made to the internal workings of APL, by changing the way variables are stored within workspaces. The interpreter can now refer to a variable in the workspace via a "reference" to it (in effect, a pointer to the value), rather than having to incur the cost of making another copy of the data. This change results in general speed-ups throughout the system, particularly for functions that manipulate enclosed arrays.

Aside from speeding up execution, the use of references (or "joint representation") decreases the space required to execute some APL expressions. First, variable assignment avoids copying the value being assigned wherever possible. Therefore, `A←B←C←20000⍴'○'` creates only one 20 000-element array rather than three. Second, calling a function with variables as arguments no longer makes a copy of the named arguments. Both of these features should improve programming style by eliminating the use of coding tricks to avoid duplication of large objects in the workspace.

Introducing joint representation into the interpreter substantially modified the internal representation of enclosed arrays. As a result, the overhead required to store an enclosed array is about 80 per cent less.

More information on joint representation may be found in this issue's *Technical Supplement.*

Dual and composition operators improved

Simplifications resulting from the use of joint representation have sped up the dual (¨) and composition (ö and ⍤) operators substantially. Both dual and composition with disclose as the right function run up to three times faster than they used to.

Extensions to event trapping

A number of extensions were made to SHARP APL's event-trapping facilities. Action code *D* now extends to all events; previously it was restricted to event 2001 (return to immediate execution). Thus you can request that the workspace be cleared if an error occurs, using the trap definition `0 D CLEAR`. When `D EXIT` is used, the system reacts by propagating the event up one stack level and signalling the event in the calling function's environment. This "resignalled" event may be trapped by another trap expression, or possibly by the same *D* trap (in which case the cutback and signalling procedure is repeated).

The action codes *C*, *D*, *E*, *N*, and *S* may be used without *any* event numbers. Eliding the event numbers causes the trap expression to trap all events within the purview of the action code. That is, when action codes *C* and *E* are used without event numbers, they will trap all class-0 and -1000 events; action code *D* will trap all class-0 and -1000 events, as well as event 2001.

To enable an event trap to handle events occurring only at a particular level of the execution stack, there is a new action code, *O* (Own). Action code *O* is the first action code which behaves as a *qualifier*, and is used in conjunction with another action code. For example, the trap expression `0 O E →L2` will trap events occurring only at the level at which the associated `⎕TRAP` is localized. Using the *O* qualifier makes it possible for a function to disable its traps for functions it calls, without relying on the called functions themselves to do it.

Explicit ranges of event numbers are now permissible in trap definitions. For example, the range `1-16` will trap all events from `1` to `16` inclusive, which represents the common APL errors. Ranges are particularly useful when specifying classes of user-defined errors, as in the trap definition `1-499 600-699` *C SAVE*.

Conditioning your program environment

APL programmers have always been faced with the difficulty of writing functions which will behave robustly when users accidentally or maliciously tamper with them. *⎕TRAP* made it significantly easier to write robust, secure systems in APL, but there remained the difficulty of ensuring that a program had the chance to *set* *⎕TRAP* and other aspects of the program's environment prior to any interruption.

To handle this problem, a new system variable, *⎕EC* (Environment Condition), has been introduced. Localizing *⎕EC* essentially makes a program behave like a primitive APL function would, until you indicate otherwise. This gives the program a chance to set *⎕TRAP*, *⎕IO*, or perform any other desired actions before allowing the user to get control.

For more details on *⎕EC*, see the article in this month's *Technical Supplement.*

)*FNS*,)*VARS*, and)*GRPS* run faster

The system commands)*FNS*,)*VARS*, and)*GRPS* now use as little as four per cent of the execution time previously taken. The speed-up increases with the number of names to be printed. The new algorithm requires a certain amount of working storage in which to operate. As a result, in a workspace which has very little free space, the older, more time-consuming algorithm may still be employed by the system.

Performance of ⎕*WS* improved

Most operations performed by the system function ⎕*WS* are faster. `1` ⎕*WS* **n**, which returns identifier names by class, uses an improved sorting algorithm and runs as much as 65 per cent faster than it previously did. `3` ⎕*WS* **id**, which returns the members of a group, is about 50 per cent faster. `4` ⎕*WS* **ids**, which computes the space occupied by one or more objects, runs slightly slower due to improved error detection and proper computation of the space consumed by a group contained within a group.

Result of ⎕*NL* now sorted

⎕*NL*, which returns the names of objects in a particular class, now sorts its result alphabetically. The collating sequence used in the sort is the same as that employed by)*FNS*,)*VARS*,)*GRPS*, and ⎕*WS*, namely ⎕*AV*[⍋⎕*AV*≠' ']. Previously, the rows of the result of ⎕*NL* were returned in accidental order.

Complex arithmetic extended

More complex arithmetic functions are available. The floor (⌊), ceiling (⌈), residue (|), and representation (⊤) functions with complex arguments now give results, rather than *DOMAIN ERROR*. The latter three primitives have correlated definitions which may be expressed in terms of complex floor. The definition of floor which was implemented was proposed by E.E. McDonnell, in his paper "Complex Floor", presented at the APL 73 Congress in Copenhagen. The definition essentially preserves the property that the distance between a number and its floor be strictly less than unity.

In addition to the extension of these four primitives to the complex domain, the complex magnitude function is five times more accurate than it was.

Input acquisition during bounce

Entering ⎕ or ⍞ input acquisition in any type of task which is being bounced now results in the message *INPUT INTERRUPT* and immediate task termination. ⎕*ER* in the saved workspace will properly indicate the cause of the termination. Previously, input acquisition via ⎕ exhibited this behaviour, but ⍞ did not.

Improved *HSPRINT* and *FILESORT* user interfaces

The functions in workspaces `1` *HSPRINT* and `1` *FILESORT* have been replaced by more efficient and more compact versions. The new functions use shared variables and a system quartermaster **N**-task to manipulate their request queues, rather than performing the operations in the user's workspace. The *BTASK* and *HCPRINT* request submission systems already utilize the quartermaster task.

With an autonomous N-task, the user functions no longer need to manipulate sensitive data or site-dependent file passnumbers. As a result, the new cover functions are simple, self-contained, unlocked programs which request service from the N-task but do very little work themselves.

In addition to the improved user interface, a number of other enhancements were made to the *HSPRINT* facility. A major reworking of *HSPRINT*'s internal organization has resulted in enormous reductions in the cost of performing some inquiries. (*MINE*, for example, is several hundred times faster.) *HSPRINT* requests are processed automatically upon submission by the user, which means decreased turnaround time—particularly for requests with destination of *FILE* or *REMO*. Some categories of *HSPRINT* requests, such as those creating tapes, may still require operator intervention prior to processing.

System availability

The SHARP APL system is designed to run virtually continuously, with most forms of data back-up being performed while the system is available to timesharing users. However, sometimes failures in important but non-critical hardware components—such as certain disk drives connected to the system—can cause the system to hang or even crash.

This release of the SHARP APL product running under the MVS operating system allows both APL and the workspace back-up utility to run, even when some disks containing users' workspaces or swap extents are not available. The increased resilience permits an APL system to provide service to users who are not affected by the unavailable devices. If a user attempts to access a workspace on a lost device, the error message *WS NOT READABLE* is returned.

Back-up utilities improved

The workspace incremental back-up utility has been improved. The new utility is capable of running essentially at the speed of the tape drive to which it is writing. This typically improves the previous execution time by a factor of five.

The file incremental back-up utility now permits concurrent archiving of multiple volume classes in the MVS environment. A **volume class** is simply a portion of a file system. For in-house sites with large amounts of data to store, the file system can be partitioned into disjoint volume classes. This change to the file utility allows a number of these classes to be backed up at the same time, thereby reducing how long it takes over all to run file back-up.

Site-dependent system exits

To increase the control that an in-house site has over the SHARP APL product, the system now contains a number of internal exit points at which a site may introduce its own customized behaviour. As an example, in the new release, the format and composition of the system sign-on and sign-off reports are under the complete control of each site. A number of other exits were added throughout the system, and more will be introduced in future releases.

VSAMAP and FCAP: two new auxiliary processors

An **auxiliary processor** is a program that runs on the same computer as APL, and communicates with APL users by means of shared variables. SHARP APL supports two new auxiliary processors, Virtual Storage Access Method Auxiliary Processor (VSAMAP) and Function Call Auxiliary Processor (FCAP). The version of VSAMAP available from I.P. Sharp is a new program whose behaviour is upwardly compatible with IBM's AP 123. The auxiliary processor is capable of reading, writing, and updating VSAM key-sequenced and entry-sequenced data sets. Access to use VSAMAP is controlled by a special system user table.

The FCAP processor, as the name implies, can be used to call non-APL programs from an APL task. FCAP will access a subroutine from a particular library, convert the APL user's data to the format expected by the subroutine, invoke the subroutine, and finally return its result to the user. Subroutine argument specifications and access controls for FCAP are also controlled by a user table.

Crosstabulations with XTABS

Yudi Plonka, Toronto

I.P. Sharp's crosstabulation package, XTABS, analyses and manipulates questionnaire data efficiently, and produces a variety of specialized reports. Market researchers, social scientists, government survey boards, and statisticians involved with the interpretation of such data will find XTABS easy to learn, powerful, and cost-effective.

XTABS contains a full range of straightforward English-type commands for handling discrete and continuous data, running reports interactively and in batch, obtaining highspeed printouts, and storing results for further analysis or revision.

Survey data is extracted from an input source to form a survey data base. Sets of data, called **variables**, are defined once and automatically saved in the data base. XTABS thus saves you a great deal of time and effort by requiring you to define variables only once instead of for each report.

Several commands are available for choosing report options and therewith generating reports that are characterized by a high degree of detail and sophistication. Some of the options included are:

- One-way holecounts to multidimensional crosstabulations
- Seven different types of totals and associated percentages
- Statistics such as mean, standard deviation, chi-square, and probabilistic measures
- Specification of a restricted sample base
- Weighting frequencies with the corresponding raw scores or scaling frequencies with row factors
- Multilevel titles and headings
- Pagination and flexible formatting features including specifications for display and rounding of significant digits, column widths, and banner breaks

Using XTABS

XTABS is not a computer language. It has been designed with the end user in mind so you don't need any computer experience to use it. All commands are available in the workspace `31 XTABS`. To begin using XTABS, simply type:

```
)LOAD 31 XTABS
```

All you really need to produce a report is a survey data base and some input data. You could create a survey data base, use the demonstration input data to add a variable called *AGE* (with appropriate headings) and produce a one-way holecount as in figure 1.

A sample data base, *MR. WAFFLE MARKET SURVEY*, is also accessible from this workspace with the *TIEDEMO* command.

In the demonstration survey data base, you can find the same variable *AGE* used in the preceding report, as well as other variables. For example, *OVERALLRANK* gives you the overall ranking of Mr. Waffle's restaurant.

You may want to experiment with the demonstration first, in order to get into the swing of XTABS. Although XTABS will not allow you to add to or enhance this demonstration data base, you still have access to all the commands necessary for displaying definitions and running reports.

XTABS contains defaults for the various options available. These defaults can be regarded as automatic report specifications for such things as the calculation of numbers, the format of titles, and the control of page depths and widths. If you don't want the control offered by a particular option, you don't have to learn how to use it. But as your needs become more sophisticated, you will find adequate provision for overriding defaults through further commands.

Contact your I.P. Sharp representative for your copy of the XTABS manual or for assistance in using XTABS to solve your survey problems.

```
      'DEMO SURVEY' CREATESURVEY 1 2000
      XTDEMOFILE INPUT 'SEQ'
      'AGE' ADDVARIABLE '12-13=G(8-15),G(16-25),G(26-35),G(36-55)'
NTASK 6765 PROCESSING....
      'AGE' HEADINGS '8-15 YEARS,16-25 YEARS,26-35 YEARS,36-55 YEARS'
      TABULATE 'AGE'
      RUN
NTASK 6765 PROCESSING....
ALIGN PAPER. PRESS RETURN KEY TO START


DATE: SEP 24, 1982                                        PAGE    1
REPORT 'CURRENT'             DEMO SURVEY

VARIABLE        HEADINGS                                 COUNT   o/o

AGE                8-15 YEARS                              279  13.95
                   16-25 YEARS                             443  22.15
                   26-35 YEARS                             650  32.50
                   36-55 YEARS                             628  31.40
                                                        -------
                                          BASE TOTAL      2000
```

Figure 1

New Petrochemical Data Base

Sally Drew, Paris

I.P. Sharp's involvement in the energy field has recently been extended to include a new petrochemical data base, ICIS. ICIS contains worldwide market information for a range of chemical and petrochemical products as reported by Independent Chemical Information Services Ltd. (ICIS). The data base is updated weekly with information relating to market activity during the previous week.

ICIS specializes in objective market research in the chemical and petrochemical industries. With more than eighty per cent of all major producers and consumers contributing information as well as subscribing to the service, ICIS has become highly respected as an industry reference.

The ICIS data base contains information on two product groups. The following list of products will likely increase early in 1983.

Chemical Group 1
butadiene, ethylene, propylene, crude C4, phenol, acetone, mono ethylene glycol (MEG), methyl tertiary butyl ether (MTBE), raffinate 1, and methanol

Chemical Group 2
caustic soda—liquid, solid, flakes, and pearls, soda ash—light and dense, sodium sulphate, vinyl chloride monomer (VCM), ethylene dichloride (EDC), and polyvinyl chloride (PVC)

The information is available in two forms via the SHARP APL Service: either as a written report in which quantities and spot and contract prices are quoted for deals concluded during the previous week, followed by a short paragraph giving market trends; or as numeric data extracted directly from the text.

You can gain access to the ICIS data base in two ways: through 121 *ICIS* or through 121 *PETROSERIES*.

121 *ICIS*

With this interactive system you can display the weekly report, tables, and graphs without any prior knowledge of computers. Online assistance is available throughout the system if you are uncertain of the valid responses to the prompts. The prompts are normally at two levels: the first is fairly short, designed to serve as a quick reminder; the second, more detailed. After you load the ICIS workspace, update information is printed. In the following examples, the latest update is assumed to be the end of August 1982. *LATEST* always gives you the most recent data.

If you had selected *REPORT* at the *OPTION* prompt, the butadiene section of each of the reports from August 1982 to the latest update would have been printed.

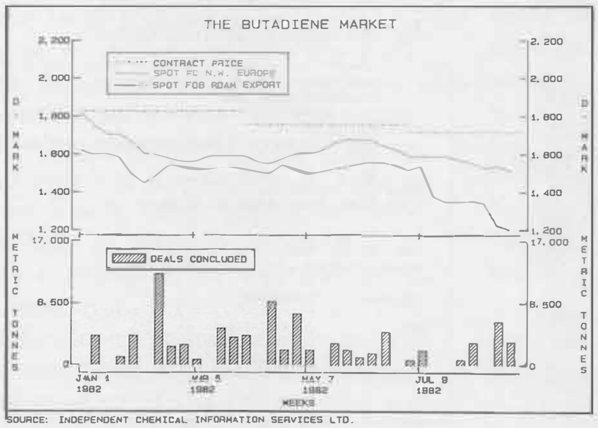
If you select the *GRAPH* option, you have the choice of printing the graph at your terminal or at your nearest I.P. Sharp office.

121 *PETROSERIES*

This workspace contains a full set of MAGIC functions (I.P. Sharp's language for manipulating time series data). With MAGIC, you can quickly and easily retrieve, analyse, and report numeric data from this data base (and from other energy or non-energy data bases available on the SHARP APL Service).

Figure 1 is a graphical summary of the butadiene market for the first eight months of 1982. The spot FOB Rotterdam export price has been converted from U.S. dollars to Deutsche Mark using the CURRENCY data base.

For further information on the ICIS data base and for access to the data without surcharge for a month, contact your local I.P. Sharp office.



SOURCE: INDEPENDENT CHEMICAL INFORMATION SERVICES LTD.

Figure 1

Volume 10/Number 6

November/December 1982

I.P. Sharp Associates
NEWSLETTER

technical supplement 41

Searching, Part IV

Robert Metzger, Rochester

This article on searching files is continued from *Technical Supplement 40.*

Directory search of a sorted file

The techniques presented so far actually require searching the file to find the record. The remaining methods reduce the work required to find a record by doing the searching in the active workspace.

It would be convenient if we could look up the record we want in some sort of directory or index. One common type of directory keeps a separate entry for each distinct key. If the records all have unique keys, however, the directory would have as many entries as records. This is usually impractical in an application which has lots of records.

If we sort the file, we can use another type of directory. In this directory we keep a record of the lowest and highest keys which occur in each component. Thus we will have two numbers per file component in our directory. This should fit in the workspace easily.

The following function implements this approach. If you keep the directory resident in the active workspace, the only file reads done will be those that actually retrieve the data you want.

```
     ∇ RECORDS←KEY FILESEARCH4 CONTROL
      ;TIE;COMPONENT;BLOCK;FIELD
[1]    TIE←CONTROL[1]
[2]    FIELD←CONTROL[2]
[3]    RECORDS← 0 0 ρ0
[4]    COMPONENT←((KEY≥INDEX[;1])
       ∧KEY≤INDEX[;2])/ι1↑ρINDEX
[5]    →(0=ρCOMPONENT)ρEXIT
[6]    BLOCK←⎕READ TIE,COMPONENT
[7]    RECORDS←(BLOCK[;FIELD]=KEY)⌿BLOCK
[8]   EXIT: ∇
```

When should you use a directory search like this? You must be able to keep the records sorted on the field represented by the directory. You must also build the directory when the file is created, and maintain it as records are added or deleted. If this is possible, this approach can be very effective.

Sequential search of a buffered file

So far, we have looked at techniques which store all of the records in the file. The next technique stores duplicate copies of certain records in the active workspace.

A buffer is an area which separates two regions. When we buffer input or output in a computer program, we store data in a temporary location which is neither its source nor its destination. When we buffer records from a file, we store them in a global variable in the workspace.

The records which are buffered are those which have been used most recently. If these records are used more than once, the file is not searched because they are already in the workspace. The assumption is that those records which have been used before are the most likely to be used again soon. Since there is only so much space in the workspace, the number of such records must be limited.

The following function implements this technique. Two types of enhancements could be made to it. First, it is only buffering reads. If records are to be modified, these too can be buffered. If multiple users can simultaneously update the file, such buffering becomes more complicated. Second, more sophisticated means can be used to determine which records to keep in the workspace. One method would keep a reference count for each record in the buffer. When the buffer gets full, the records with the lowest reference counts would be removed from the buffer. The reference count might then be set back to 0.

```
      ∇ RECORDS←KEY FILESEARCH5 CONTROL
        ;FIELD;TIE
[1]     TIE←CONTROL[1]
[2]     FIELD←CONTROL[2]
[3]     RECORDS← 0 0 ρ0
[4]     →(~KEY∊BUFFER[;FIELD])ρNOTINWS
[5]     RECORDS←(BUFFER[;FIELD]=KEY)
        ⌿BUFFER
[6]     →EXIT
[7]   NOTINWS:RECORDS←KEY FILESEARCH1
        CONTROL
[8]     BUFFER←RECORDS,[1] BUFFER
[9]     BUFFER←((MAXBUFFERSIZE⌊
        1↑ρBUFFER),¯1↑ρBUFFER)↑BUFFER
[10]  EXIT: ∇
```

When should you use a buffered approach to file searching? If a relatively small number of records gets most of the usage, it can be a very effective way of minimizing file searching costs.

Algorithmic search of a hashed file

The final search method really doesn't involve searching at all. It computes the component number that the record should be in, and reads that component directly.

The method is called "hashing" because files set up this way are neither in sorted order nor in random order, but rather scrambled in a very special way. The basic idea is to find a calculation which can be performed on the key to produce the appropriate component number.

There are a number of constraints on choosing the calculation. It must be simple, so that it is less expensive than the other methods. It must also distribute the keys relatively uniformly.

Once the algorithm for calculating the location has been selected, some other design choices must be made. The maximum number of records per component must be high enough to make overflow infrequent. In addition, a method must be chosen to handle overflow when it does occur.

The algorithm most often chosen for hashing is to take the remainder after dividing the key by a prime number. (Chapter 21 of *Computer Data Base Organization* by James Martin explains other hashing algorithms, and how their performance compares to this one.) In APL, this algorithm is simply `1+HASHNUMBER|KEY`.

The function listed below implements this type of search. Note that it presumes the existence of two global variables: HASHNUM and MAXRECORDS. The file must start with HASHNUM components. When a block has MAXRECORDS records in it, any extra records which have keys which hash to this location will be stored sequentially in the components following HASHNUM.

```
      ∇ RECORDS←KEY FILESEARCH6 CONTROL
        ;TIE;FIELD;BLOCK;CTR;LMT
[1]     TIE←CONTROL[1]
[2]     FIELD←CONTROL[2]
[3]     RECORDS← 0 0 ρ0
[4]     BLOCK←⎕READ TIE,1+HASHNUM|KEY
[5]     →(~KEY∊BLOCK[;FIELD])ρOVERFLOW
[6]     RECORDS←(BLOCK[;FIELD]=KEY)⌿BLOCK
[7]     →EXIT
[8]   OVERFLOW:→(MAXRECORDS>1↑ρBLOCK)ρEXIT
[9]     CTR←HASHNUM
[10]    LMT←¯1+1↑1↓⎕SIZE TIE
[11]  LOOP:CTR←CTR+1
[12]    →(LMT<CTR)ρEXIT
[13]    BLOCK←⎕READ TIE,CTR
[14]    →(~KEY∊BLOCK[;FIELD])ρLOOP
[15]    RECORDS←(BLOCK[;FIELD]=KEY)⌿BLOCK
[16]  EXIT: ∇
```

When should you use hashing to locate records? You must be able to find an algorithm (just a hash number, if you use the residue approach) which distributes the keys relatively uniformly over the component numbers. If you don't, the application will be slowed down by excessive sequential searching of the overflow areas. You must have a relatively even distribution of records over keys, if it is possible for more than one record to have the same key. Otherwise, you may suffer from *WS FULL*s. You must have a relatively stable number of records. If not, you may incur excessive costs due to frequent file reorganizations. If your data meets these requirements, hashing can be a very efficient means of accessing data stored on file.

Comparing approaches

A few general comparisons may be helpful. The directory and hashed approaches require a determination of what field can be searched when the file is designed. The inverted and binary search approaches require substantial effort to organize the file. The sequential and buffered approaches require neither prior determination of search fields nor substantial preparation of the file, but they are only effective under certain special usage patterns. When you design a file, you must take all these factors into account.

Entire books have been written about file searching. What we have done here is present a sampler of ideas and algorithms. We hope that you can use them to meet your file searching needs in APL applications. If you need these programs in your applications, you can find them in workspace 777 *SEARCHING*.

Acknowledgements: My thanks to J. Henri Schueler for his assistance in preparing this article.

Joint Representation

Robert Bernecky, Toronto

One problem encountered by APL users is *WS FULL* at function call time. Well-designed systems using defined functions with formal arguments were especially susceptible to this, as application size grew faster than workspace size. Circumvention of the problem drove users to such drastic measures as passing globals to functions, or avoiding the use of a function altogether. Such techniques complicate system maintenance and invite insidious bugs.

The November 1982 release of SHARP APL alleviates this problem, provides substantial storage and processor time improvements for enclosed arrays, and provides a number of other performance enhancements by the introduction of a scheme for workspace storage management called **joint representation**.

Joint representation is essentially the "slack representation" technique described in "Representations for Enclosed Arrays" [1], applied to all arrays, simple as well as enclosed, in the workspace. Joint representation allows multiple references to the same value array. A reference count associated with each value is used to indicate when an array may be discarded. The sharing is totally controlled by the APL interpreter, and the user need not be cognizant of its existence. Multiple references may occur for several reasons:

- Assignment
- Function call
- Primitive function identities
- As a side effect of the enclose function

Assignment

Previously, *X*←*Y*←⍳5 would have created two copies of the array ⍳5, assigning one copy to *X*, and one copy to *Y*. The requirement for separate copies used workspace storage and processor time directly proportional to the amount of storage occupied by the right argument. With joint representation, only one copy of the array ⍳5 would be created, and its value would be shared by both *X* and *Y*. Clearly, an indexed assignment into *X* or *Y* still has to make a copy, so that the other variable doesn't change value. An interesting example of joint representation is that two variables may have their values exchanged without *WS FULL*, in fixed time, regardless of the size of the variable, with no data movement:

```
        X←(⌊.4×⎕WA)⍴'X'
        Y←(⍴X)⍴'Y'
        T←X ⋄ X←Y ⋄ Y←T
```

Function call

Function call is similar to assignment, in that a formal parameter is assigned the same value as the corresponding function argument. As with assignment, this previously took time and space proportional to the size of the argument(s). With joint representation, the time and space requirements are small and independent of the argument size. This allows application designers to write defined functions which don't depend on global variables or side effects, and which won't stop working when the application grows in size. Joint representation should encourage the use of modular code and functions which may be used freely as building blocks for new applications.

```
    ∇ R←RECURSE ⍵
[1]   R←1+RECURSE ⍵
    ∇
      ⎕TRAP←'/1 E →R←0' ⍝ MAXIMUM RECURSION DEPTH.
      RECURSE 100000⍴⎕AV
3842
```

Primitive function identities

Many primitive functions have special cases which produce an unaltered copy of the right or left argument as the result. Some of these may be easily detected in the course of execution (e.g. ravel of a vector). Others are more difficult or impractical to detect, such as:

```
        X[4]←X[4]
```

SHARP APL detects certain identities, and returns a reference to the argument value as the result, instead of making a copy of the argument. Since the argument isn't physically copied to produce the result, the time and space requirements are independent of the argument size:

```
      )CLEAR
      Z←A←250000⍴⎕AV
      ⎕WA
51460
      B←A
      ⎕WA
51472
```

The growth in ⎕WA is due to the difference in length of the previous immediate execution lines, which are preserved in the workspace for recall by the line editor. The following examples are all nilpotent, in that the explicit result of the function is one of the arguments:

```
        C←,B
        D←(⍴C)⍴B
        E←(⍴D)↑C
        F←0↓E
        G←⍉E
        H←(⍴E)⌽G
        I←1\H
        J←((⍴H)⍴1)/I
        K←J[]
        L←K,⍳0
        M←'',L
        N←⍕M
        O←>N
```

Clearly, some of the above examples are rarely encountered, and of little practical use. However, they are usually detected in the normal course of conformability checking, and do not measurably impact the performance of the interpreter.

Enclosed arrays

Joint representation also has been applied to enclosed arrays, replacing the coadunate representation strategy previously used. The result is a significant reduction in processor time and storage costs associated with the use of enclosed arrays. In particular, the cost of creation of enclosed arrays, or of materializing them in the workspace via such methods as ⎕*READ*, ⎕*PDEF*, or shared variable reference has been drastically reduced. Joint representation, combined with the optimization of the composition operators also appearing in the November 1982 release of SHARP APL, has resulted in cost savings of 15 to 40 per cent for some systems which rely heavily on the use of enclosed arrays.

Joint representation also lets enclosed arrays share values with non-enclosed arrays, by allowing both enclosed and non-enclosed values to refer to the same array:

```
        P←<O
```

Since the enclosed value is a reference, so is the result of disclose, if the array being disclosed is a scalar:

```
        Q←>P
```

Anomalies corrected

By stacking references to variables at the time they are encountered by the syntax analyzer, joint representation removes the anomalies which were observable when the variable name was stacked instead:

```
      (A←2)+A←5 ⍝ RESULT IN PREVIOUS RELEASES WAS 4
7
      A
2
      (⎕EX'A')+A←5 ⍝ WAS NONCE ERROR
6
      A
VALUE ERROR
      A
      ^

      A←2;A←5 ⍝ WAS 22
25
      A
2
```

Footnotes

[1] Bernecky, Robert. "Representations for Enclosed Arrays," *APL 81 Conference Proceedings, APL Quote Quad*, Vol. 12, No. 1 (September 1981).

New System Variable

Leslie Goldsmith, Toronto

A new system variable, ⎕*EC* (Environment Condition), has been added to the SHARP APL system. ⎕*EC* is designed to simplify the writing of secure or sensitive programs in SHARP APL by enabling a program to react, exactly as a primitive function would, if an error or an asynchronous event (such as ATTN) occurred within it.

⎕*EC* is a Boolean system shared variable. The global default of the variable is 0. When localized, ⎕*EC* assumes an initial value of 1. Assigning an improper value to ⎕*EC* always effects the default setting 0.

In the absence of event traps to handle errors and other events, the effect of ⎕*EC* is as follows. When the variable is everywhere 0 on the state indicator (SI) stack, the system takes its standard, default action when an event occurs.

When the variable is 1, that particular stack level, and those more local than it, are designated to be secure. Then the system takes the action listed below, depending on the actual event:

- *Errors* cause the stack to be cut back one level, and the error to be signalled in the environment of the caller. If this environment is also secure (that is, ⎕*EC* is still 1), then the procedure is repeated.
- *Single* ATTN is ignored, but is left pending. The ATTN will be honoured as soon as possible after ⎕*EC* reverts to a non-secure setting.
- *Double* ATTN (interrupt) and other asynchronous events are treated as errors.
- *Stop control* and *return to immediate execution* behave as in a non-secure environment.

In other words, errors and interrupts are passed on to the most local level which does not have ⎕*EC* set to 1. When an untrapped error or interrupt occurs, the user is *never* given control while there is a latent 0 value of ⎕*EC* anywhere on the stack. Therefore, by localizing ⎕*EC* and not redefining it anywhere within the body of the program, the program will behave exactly the way an APL primitive function would.

Stop control is honoured even when ⎕*EC* is on as a convenience and a debugging aid to the programmer. Note that, as was previously the case, stop (and trace) control may not be altered once a function is locked.

Because single ATTN is masked off by a secure setting of ⎕*EC*, two ATTNS are required to terminate ⎕*DL* while ⎕*EC* is 1. ⎕*EC* effectively makes ⎕*DL* behave as a primitive function in this case.

Interaction of ⎕*EC* and ⎕*TRAP*

Event traps may be used in conjunction with secure environments, so that a function can be made to behave as a primitive only if the event is not trapped. Where there is the possibility of interaction of ⎕*TRAP* with ⎕*EC*, the system proceeds according to the following rules. When an event occurs, the levels of the SI stack are examined, working from the most local level to the global one. At any level, if a value of ⎕*TRAP* which handles the event *and* is localized at that level is found, then that trap is given control. If not, then if ⎕*EC* is localized at that level, the setting of ⎕*EC* is acted upon: a value of 0 has no effect, and the stack search continues. A value of 1 causes the behaviour described above. These rules reduce to the simpler ones given earlier if ⎕*TRAP* does not come into play.

The system reverts to its default action if neither ⎕*TRAP* nor ⎕*EC* at any level of the SI stack acts upon a particular event. The nature of this search ensures that programs will continue to operate as they have in the past, even with the introduction of ⎕*EC*.

Using ⎕*EC*

⎕*EC* now makes it possible for a program author to ensure that a program has a chance to condition its environment (set event traps, origin, and so on) before any interruption can occur. For example:

```
     ∇ PASSCHECK;⎕EC;⎕IO;⎕TRAP
[1]    ⎕IO←0 ◇ ⎕TRAP←'∘ 19 C →ACCERR ∘ D EXIT ∘ S'
[2]    →(ρ⎕TRAP)↓0 ⍝ EXIT IF ⎕TRAP INVALID FOR SOME REASON
[3]    ⎕EC←0 ⍝ ENVIRONMENT NOW SAFELY CONDITIONED
[4]    ...
     ∇
```

Because ⎕*EC* assumes a fail-safe default when it is localized, no action on the user's part can alter the author's intended program flow. Since localization information is contained within the static header of the function, the efficacy of the mechanism does not rely upon the dynamic execution of certain parts of the program. This is a fundamental distinction between the protection facilities provided by ⎕*EC* and those provided by ⎕*TRAP*.

The dynamic nature of ⎕*EC* also permits it to be set based on the result of some expression. For example, the following statement will mark the environment safely conditioned only if the user executing the program is in a list of trusted users:

```
[1]    ⎕EC←~(1ρ⎕AI)∊ 1618033 2718281 3141592
```

⎕*EC* can be used more elaborately, to condition critical sections of a program dynamically. Within a section where ⎕*EC* is 1, the program will react like a primitive and weak ATTN will be ignored (unless it is trapped). At the completion of the section, setting ⎕*EC* to 0 will turn this behaviour off. For example:

```
     ∇ TRANSMIT;LC;INP;TXT;⎕EC;⎕IO;⎕TRAP
[1]   ⍝ LOGS A STATUS MESSAGE FOR PERUSAL BY THE
       SYSTEM STEWARD.
[2]   ⍝ THE SYSTEM FILE IS ASSUMED TO BE TIED TO THE
       GLOBAL VARIABLE
[3]   ⍝ <TIE>.
[4]   ⍝
[5]   ⍝ OTHER GLOBALS:  V - CR;  F - TAD
[6]   ⍝
[7]    ⎕EC←⎕IO←0 ⍝ ENVIRONMENT REQUIRES NO SPECIAL
       CONDITIONING YET
[8]    'TEXT:' ◇ TXT←'' ⍝ PREPARE FOR TEXT COLLECTION LOOP
[9]   MORE:→(' '∧.=INP←⍞)ρDONE ⍝ STOP ON <CR> OR <SPACE,CR>
[10]   TXT←TXT,INP,CR ◇ →MORE ⍝ ACCUMULATE LAST INPUT LINE
[11]  DONE:⎕EC←1 ⍝ CRITICAL SECTION STARTS HERE...
[12]   ⎕TRAP←'∘ 21 E LC←⎕LC ◇ →FFULL' ⍝ TRAP FOR 'FILE FULL'
[13]   TXT←'STATUS MESSAGE AS OF ',(TAD ⎕TS),CR,CR,TXT
[14]   ⎕FHOLD TIE ⍝ OBTAIN FILE INTERLOCK FOR COMING WRITES
[15]   TXT ⎕APPEND TIE ⍝ APPEND STATUS REPORT TO SYSTEM FILE
[16]   ((⎕READ TIE,5),[0](1ρ⎕AI),1ρ⎕RUNS) ⎕REPLACE TIE,5
       ⍝ AUGMENT DIRECTORY WITH OUR ACCOUNT NUMBER AND TASKID
[17]   ⎕FHOLD '' ◇ 'STATUS REPORT FILED'
[18]   →0 ⍝ RELEASE HOLD BEFORE RETURNING
[19]  FFULL:(50000+1ρ2↓⎕SIZE TIE) ⎕RESIZE TIE
[20]   →LC ⍝ FILE FULL; RESIZE AND RETRY
     ∇
```

This program files a status report entered by the user, and also records some ancillary information in a directory. Prior to the sensitive section of code, *TRANSMIT* collects the message text from the user. Since strict security is not the issue in the program, it is unnecessary for this step to be performed in a secure environment, and in fact preferable that it is not. If *⎕EC* were 1 during text collection, then any error or interrupt (including a line drop) would cut back the stack, thus losing any text the user had entered. On the other hand, this behaviour is very desirable during the update section of *TRANSMIT*, where permitting execution to be interrupted and resumed would destroy the program's file hold. Although one can sometimes handle the division of environments by placing critical sections in subroutines, there are often overriding reasons why a program must be self-contained.

Summary

⎕EC is a powerful, yet simple-to-use programming facility, which complements the protection features offered by *⎕TRAP*. *⎕EC*'s fail-safe handling of errors, and its ability to mask ATTN dynamically, enormously simplify the task of writing secure software in SHARP APL.

Workspace of the Month

7 WSSEARCH

Clement Kent, Toronto

WSSEARCH joins the group of tools in library 7 for manipulating and documenting programs and workspaces. *WSSEARCH* provides programs to search some or all of the functions in a workspace for specific strings, and to replace occurrences of those strings by new values. *WSSEARCH* programs accept a variety of "flags" to specify special actions (return output as explicit result; search comments, numeric vectors, quoted strings, or strings following ⍎; display messages about locked functions; and much more).

Programs in 7 *WSSEARCH* fall into two groups. Those prefaced by *FN* (*FNFIND*, *FNSHOW*, *FNRPLC*, *FNREPLACE*) perform context-free searches. *Any* occurrence of the string, whether it be in a comment, quoted string or part of a variable name, will be located. Functions prefaced by *SE* (e.g. *SESHOW*) search for syntactic elements. Suppose, for example, you wanted to see all instances of the variable *A* in the function *CAT* below:

```
    ∇ R←A CAT B
[1]  ⍝ ADD VECTOR B AS ROW TO MATRIX A
[2]   R←(R↑A),[⎕IO](¯1↑R←(ρA)⌈0,ρB)↑B←,B
    ∇
```

You could try to show them with *FNSHOW*:

```
      'CAT' FNSHOW 'A'
SEARCHING 1 FUNCTION

    ∇ R←A CAT B  (8 OCCURRENCES)
    ∇ R←A CAT B
        ^  ^
[1]  ⍝ ADD VECTOR B AS ROW TO MATRIX A
       ^            ^          ^     ^
[2]   R←(R↑A),[⎕IO](¯1↑R←(ρA)⌈0,ρB)↑B←,B
           ^               ^
```

Or you could show only the occurrences of *A* as a syntactic element:

```
      'CAT' SESHOW 'A'
SEARCHING 1 FUNCTION

    ∇ R←A CAT B  (3 OCCURRENCES)
    ∇ R←A CAT B
        ^
[2]   R←(R↑A),[⎕IO](¯1↑R←(ρA)⌈0,ρB)↑B←,B
           ^               ^
```

You can see from the examples that searching for *A* as a syntactic element yields only valid occurrences of the variable *A*, while the context-free search finds all the occurrences of the letter *A*.

The right argument to *WSSEARCH* programs specifies the strings to be searched for, while the left argument names the functions (or variables) to be searched. An empty left argument means that the program is to search the entire workspace. To do multiple simultaneous searches, join the disparate strings with the utility function *AND*:

```
      'CAT' SESHOW '⎕IO' AND '⌈' AND '¯1'
SEARCHING 1 FUNCTION
   ∇ R←A CAT B  (3 OCCURRENCES)

[2] · R←(R↑A),[⎕IO](¯1↑R←(⍴A)⌈0,⍴B)↑B←,B
                  ^   ^        ^
```

An important feature of *WSSEARCH* gives you the ability to search for occurrences of "old" strings and replace them by "new" strings in one step. All replacements are done simultaneously to avoid confusion:

```
      '⍝CAT' SERPLC  'A'BY'B'  AND  'B'BY'A'
           AND  'CAT'BY'ADD'
SEARCHING 1 FUNCTION
10 REPLACEMENTS MADE.
     ∇ADD[⎕]∇
   ∇ R←B ADD A
[1]  ⍝ ADD VECTOR A AS ROW TO MATRIX B
[2]   R←(R↑B),[⎕IO](¯1↑R←(⍴B)⌈0,⍴A)↑A←,A
   ∇
```

You can change the name of the function just like any other syntactic element. The variables *A* and *B* have been switched properly, even in the comment. The comment flag ⍝ in the left argument directed that all syntactic elements in the comment be searched as well.

All functions in 7 *WSSEARCH* are self-contained and you can copy them into your workspace. There are many other features documented on line in *DESCRIBE*, *SUMMARY*, *HOWFIND*, *HOWSHOW*, *HOWRPLC*, and *HOWREPLACE*.

You can use *WSSEARCH* to edit functions or their character vector representations. (A more interactive function editor is found in 7 *FNED*.) You can produce a cross-reference listing which finds any set of strings in any set of functions (including or excluding comments or quoted strings, using syntactic or context-free searching) using *SEFIND* or *FNFIND*. This supplements the workspace cross-reference facilities of 7 *XREF*.

WSSEARCH was designed and implemented by Leslie H. Goldsmith, and is an I.P. Sharp proprietary software product.

Drawing APL Trees: **Answers to Brain-Teasers**

Clement Kent, Toronto

In the last *Technical Supplement* I posed several problems in my article "APL Trees". Here are the solutions.

The first problem concerned the size and shape of some dead stick trees. All these problems can be solved if you use the formula for the sum of a geometric series. For instance, I asked how you would prove that a tree with basic branch 1 by 1/3 units, and shrinking factor 1/3, fits into a box 1.5 by 1 units. Well, the series z+z/3+z/9+ . . . sums to 3z/2, so when z=1 (abscissa) the box is 1.5 units, and when z=1/3, each half of the tree is 1/2 unit, so the whole tree is 1 unit thick.

What's the total length of all the branches? The basic branch has length *Z*←|1*J*÷3, which Pythagoras tells us is (square root 10)/3. Each time you go up one level in the tree, the number of branches doubles but their length shrinks by 1/3. So now the series is 2(z+2/3z+4/9z+ . . .)=6z. The total length is the square root of 40. I tried to mislead some of you with dark hints about twice pi, which at 6.28 is indeed very close to the true length 6.32, but not the same. I asked how many levels up you would have to go before the length exceeded twice pi . . . that's slightly harder, but still quite possible. The sum +/2×*Z*×(2÷3)*0,⍳*N* is 6×*Z*×1-(2÷3)**N*+1. So a little algebra shows that *N*=⌈((⍟1-○10*-÷2)÷⍟2÷3)-1, which is 12.

Last but not least, I asked how you would find the angles made by tangents to a logspiral branch. The equation for the spiral as a function of time T was *B*←*ω×*T* where ω was a complex number combining the shrinkage and rotation rates for the spiral. Now, if you're a calculus fan, you saved yourself a lot of time and agony because you remembered (didn't you?) that the derivative of an exponential is: *DB*←ω×*B*.

We can normalize DB so each of its members has magnitude 1 by doing *DB*←×*DB*.

We can now translate B to the origin (*B*←*B*-*B*[1]), rotate it "flat" against the real axis (*B*←*B*÷*DB*[1]) and the end will make an angle *ANG*←12○ *DB*[⍴*DB*]÷*DB*[1].

CHANGES

Nelly Detre, Paris

Daily buying and selling rates, in French centimes (1/100 of a franc), for 21 currencies are now available in the Currency Exchange Rates (CURRENCY) data base. The Banque Française du Commerce Exterieur (B.F.C.E.) updates the rates at approximately 17:00 UTC on the day of reporting. Historic data going back to 1961 was supplied by the B.F.C.E.

To help users access the information on the official Paris market, an interactive retrieval system, with prompts in French, is available in the workspace 772 *CHANGES*. Companies involved with accounting, fiscal and customs declarations, market studies and so forth, with other countries, will find *CHANGES* extremely useful in determining foreign exchange rates.

CHANGES processes the buying, selling, and fixing daily rates on a monthly, quarterly, semiannual, or yearly basis. The rates can be calculated on the average, or minimum or maximum value. The nearest daily rates, for example, for weekends, holidays, and the last day of the month are also available.

The prompts are formulated for the currencies with a three-letter code, in accordance with the French standard (AFNOR K10-020) and the International Standards Organization (ISO 4217). You can obtain the information as a table or graph. The graphs are plotted with SUPERPLOT. Dates or periods may be continuous or discontinuous.

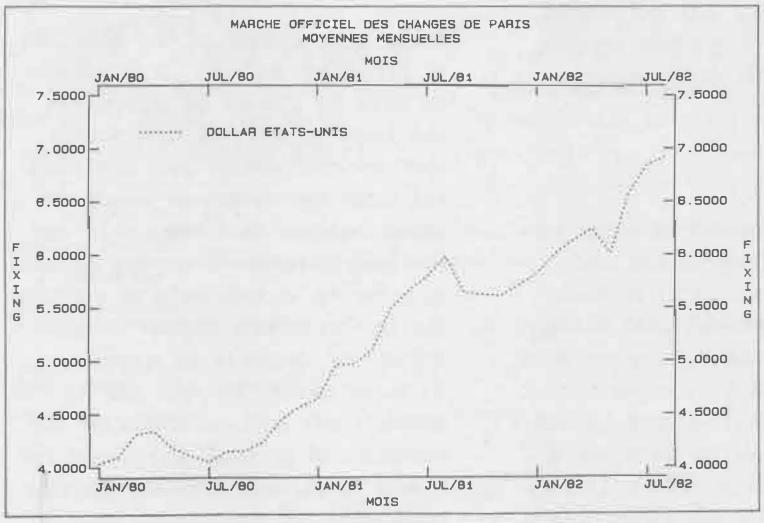
For more information and for the brochure on *CHANGES*, contact the I.P. Sharp office in Paris.

The following table shows the minimum buying and maximum selling rates for the U.S. dollar, Deutsche Mark, and Japanese yen for the first quarters of 1981 and 1982, and the second quarters of 1981 and 1982. A graphic summary of monthly averages of fixing rates for the U.S. dollar is shown in the SUPERPLOT for the period January 1980 to August 1982.

```
      )LOAD 772 CHANGES
SAVED  9.01.52 10/25/82
SOURCE :BANQUE FRANCAISE DU COMMERCE EXTERIEUR
DONNEES DISPONIBLES DEPUIS JANVIER 1961
DONNEES VALIDEES A PARTIR DE JANVIER 1973
DONNEES POUR 1 UNITE EN FRANCS
CODES DES DEVISES (OU AIDE )? :USD DEM JPY
INFORMATION SOUHAITEE (OU AIDE)? :AMINVMAXT
CODES TAB ET/OU GRA (OU AIDE) ?:TAB
DATES (OU AIDE)?  :181 182 281 282

                      MARCHE OFFICIEL DES CHANGES DE PARIS
                    MINIMUM ACHAT - MAXIMUM VENTE -TRIMESTRIELS

-----------------------------------------------------------------
                            1TRI/81   1TRI/82   2TRI/81   2TRI/82
-----------------------------------------------------------------
  DOLLAR ETATS-UNIS  MINA  4.465500  5.653000  4.944000  5.945500
                     MAXV  5.212400  6.290500  5.754500  6.924500
  DEUTSCHE MARK      MINA  2.300400  2.529900  2.353100  2.591700
                     MAXV  2.362700  2.623400  2.412200  2.780700
+ YEN JAPON          MINA  2.235000  2.523500  2.324700  2.485200
                     MAXV  2.495600  2.616800  2.562100  2.696800
-----------------------------------------------------------------
+ EN CENTIMES
```



SOURCE: BANQUE FRANCAISE DU COMMERCE EXTERIEUR

APL Terminology

Kenneth E. Iverson

I have taken your advice and the names, and use **anode, cathode, anions, cations**, *and* **ions**; *the last I shall have but little occasion for. I had some hot objections made to them here, and found myself very much in the condition of the man with his Son and Ass, who tried to please every body . . .*

Michael Faraday [1]

The many APL technical terms, coined in English, require the choice of corresponding terms when writing in, or translating to, other languages. Feeling that uniformity in these choices could be promoted by public discussion of the matter, Paul Berry suggested that the topic be broached in a session of the recent Heidelberg conference.

Professor Janko invited me to open the discussion with a talk on the considerations that have guided the choice of the terms in English. The interest in the topic expressed at the conference has persuaded me to prepare a written version, and to seize this early opportunity to publish it.

Taxonomy

The term **taxonomy** is commonly used for the classification and naming problems faced in such disciplines as geology and biology. Although the naming problems in APL are trivial by comparison, it can prove fruitful to seek guidance from taxonomists in these more difficult disciplines. Thus, D.S. Jordan [2] states the main reason for care in coining terms:

In taxonomy it is not nearly so important that a name be pertinent or even well chosen as that it be stable.

Jordan adds an acerbic comment on Linnaeus and the failing of his followers which is worth keeping in mind:

In changing his own established names, the father of classification set a bad example to his successors, one which they did not fail to follow.

Finally, R.W. Brown, in *The Composition of Scientific Words* [3], comments on the complementary need for consistency, and the price of achieving it:

If stability is to be expected, consistency must be the watchword. For the two are complementary; but harmonious consistency cannot be achieved without temporary confusion and inconvenience.

Stability

Some appreciation of the difficulty of achieving stability in terminology may be gained by comparing the large number of new words that are introduced into any natural language each year, with the small number that remain in use for long periods. If we are to aim at stability, it will help to examine the factors which appear to contribute to longevity in words. First, however, we will use an example from APL to illustrate the difficulty of gaining acceptance for a new term, and even for gaining recognition of the concept that it embodies.

The concept of an **operator** (which applies to a function or functions to produce a **derived** function, as in `+/` or `+.×`) was at the outset not recognized in APL, and the term was first used in the APLSV manual [4] in 1975. Although the distinction between a function and an operator is important in APL, speakers still frequently fail to make it, although writers seem at last to be observing it fairly consistently.

Although the term was adopted from mathematics (essentially in the sense in which Heaviside defined it), its adoption in APL has probably been retarded by the more familiar use of it in elementary mathematics and in programming languages as a synonym for **function**. If the precise usage of the term prevails in APL, it may eventually persuade mathematics teachers to use the term **function** for the elementary functions such as addition and multiplication, so that the notion of function as introduced later will no longer appear foreign or unfamiliar.

To proceed with the question of stability—survival of a word is much like survival in politics—it depends on having the right connections, and many of them. ("I have friends I haven't even used yet.") In the case of words, the connections are to similar words having related meanings or connotations; in the broad meaning of the term, they are *puns*.

Puns, both verbal and visual, both good and bad, have been widely and consciously used in APL. For example, the use of the circle (○) for the circular function is a verbal pun, and the use of ○D for the circumference of a circle of diameter D is a graphic pun (Although, as Dr. Benkard pointed out at Heidelberg, it is also a verbal pun because the symbol looks like a pie.)

The symbol ⍟ for the log function has sometimes been justified by its similarity to the appearance of the end of a log, a justification that should qualify it as a bad pun. The symbol can also be justified by a pun which is neither graphic nor verbal, but a pun *in* APL; ⍟ is justified by its relation to ⋆, the symbol for the function inverse to logarithm.

If we accept the idea that good connections (i.e. multiple supports) contribute to the survival of any new term, what general guides should one follow? In APL I know of two that have been consciously (and rather conscientiously) followed: adoption of existing terms (from other disciplines as well as from ordinary English), and examination of the etymology of proposed terms.

The adoption of existing terms is illustrated by the following list:

Mathematics :	**array, function, axis, operator**
Chemistry :	**valence**
Psychology :	**ambivalence**
Ecology :	**cline** (used by Orth [5])
Other :	**quad, ravel, stile**

Examination of the etymology of a term not only provides an estimate of the variety and suitability of the connections it possesses, but often leads one to a better term. The importance of roots may be illustrated by a completely rootless coinage that once enjoyed some currency in APL circles.

In the expression A⋆B the function ⋆ is dyadic, and in ⋆B it is monadic; to denote the characteristic in which these two uses of the function differ, the term "adicity" was used. The word is made up of three parts: *ad* (towards), *ic* (pertaining to), and *ity* (quality); all are prefixes or suffixes, and not one of them is a root.

Although coined words may not be found in a dictionary, nearby words often provide good clues to the intended meaning. Because "adicity" contains no root, the use of a dictionary will be fruitless, leading only to neighbouring words such as "adieu", and "adiabatic". However, even a bad coinage may draw attention to the need for a term; thus the use of "adicity" drove Adin Falkoff to dip into his background in chemistry to propose the term **valence**, now commonly used.

One further guide has been followed in the choice of APL terms, namely, brevity. Examples include the choice of **shape** over "dimension", **rank** over "dimensionality", and **catenate** over "concatenate".

Translation questions

In choosing APL terms, the principles which apply in English should apply equally in other languages. However, when the primary literature is in one language, there is a tendency (as exemplified by the international use of Italian phrases in music) to simply adopt terms without translation. It would seem preferable to follow the example of mathematicians in adopting the same *symbols* internationally, but (usually) translating the words for the functions they represent.

There is a second tendency which should be noted, the adoption of terms current in the computer literature; for example, the use of the German computer term "datenobjekt" as a translation of "array". Although the designers of APL have always been aware of computer terminology, few, if any, computer terms have been adopted, for the simple reason that they have appeared to be irrelevant or ill-considered.

After emphasizing the importance of puns, it is appropriate to comment on the extreme difficulty of preserving their effect in translation, and to urge, by means of an example, the marvellous effects that can be achieved by loving care. The example is one of many that can be found in an amusing and interesting book by Victor Proetz [6] on the translation of poetry. It concerns the translation of the pun on *axis* and *axes* found in the following passage from Lewis Carroll's *Alice in Wonderland*:

"You see the earth takes twenty-four hours to turn around on its **axis**—*"*

"Talking of **axes**,*" said the Duchess, "chop off her head."*

In German the pun is easily preserved:

"Was, du redest von **Axt**?" *fragte die Herzogin. "Hau ihr den Kopf ab!"*

In French, the pun is preserved, in Proetz's words, ". . . with a stroke of wizardry and judgement which, in this instance, is not translation by *word*, but by *change of word* . . .":

". . . la terre met vingt-quatre heures à faire sa **révolution**."

"Ah! vous parlez de faire des **révolutions**!" *dit la Duchesse. "Qu'on lui coupe la tête!"*

In presenting a German translation of Burns' *To a Mouse* (with correspondents such as *Mousie - Mäuschen, beastie - Tierchen,* and *housie - Häuschen),* Proetz says that "The Scottish language is perfect for addressing small children and animals. German is next because in German diminutives are familiar and homely and in everyday use." Perhaps translations to these languages can satisfy both the proponents and opponents of APL, the former because they know that the simplicity of APL makes it fit for children, and the latter because they know that APL is fit only for animals!

The question of terminology (or at least of puns) does appear to have a wide appeal, for I received many immediate comments and suggestions. In particular, Mr. A.D. Cunningham of Luxembourg gave me references to accounts of Whewell's contributions to scientific terminology [1], from which I took the epigraph used here, and Dr. D. Hirshberg of IBM Brussels gave me the following two-language aphorism:

Is life worth living? It depends on the liver!

La vie, vaut-il vivre? C'est une question de foie!

References

1. Todhunter, I. *William Whewell, D.D. - An Account of His Writings*, McMillan and Co., London, 1876, page 89.
2. Jordan, David Starr. *Guide to the Study of Fishes*, 1905, cited in [3], page 57.
3. Brown, Roland Wilbur. *Composition of Scientific Words*, published by the author, Reese Press, Baltimore, Md., 1956, page 57.
4. *APL Language*, IBM GC26-3847-4, 1978.
5. Orth, D. *Calculus in a New Key*, APL Press, Palo Alto, Ca., 1976.
6. Proetz, Victor. *The Astonishment of Words*, University of Texas Press, Austin, Texas, 1971.

APS A Planning System

Jody Davies, Sydney

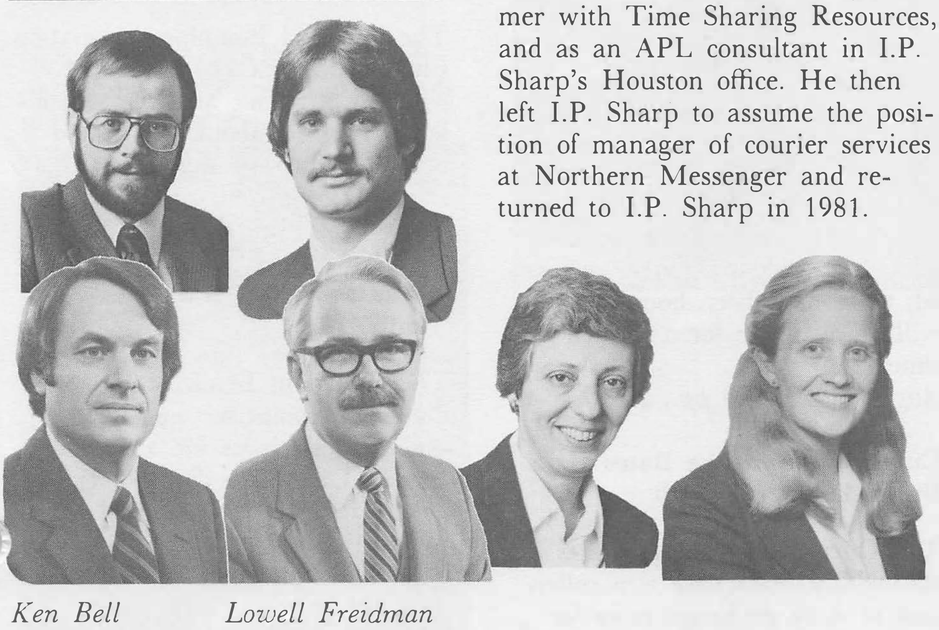
Jim Cobb, the managing director of AIWA Australia, was looking for more than automated accounting packages to manage the company efficiently. AIWA imports stereo systems and distributes them to retail stores and outlets in Australia.

Cobb came to I.P. Sharp to find a management tool, a system that would allow him to forecast accurately and simply with the control and flexibility he required. This was the impetus for developing APS (**A** **P**lanning **S**ystem) written by I.P. Sharp Australia. APS helps in the planning and control of an organization. APS incorporates a data storage and retrieval system, a report writer, and a powerful, open-ended set of calculation routines. But most importantly, APS was developed with the specific needs of a user in mind. The writing of each step required a working relationship between a customer willing to become involved and an I.P. Sharp representative. In this case, Paul Phillips of I.P. Sharp was an essential ingredient in the creation of this effective package.

"I cannot recommend the system highly enough," Jim Cobb says, since with APS he can do sales and product forecasting and easily compare actual figures to predictions. When sales for an AIWA stereo system fell below forecasted figures in April of this year, AIWA had to determine an optimal reduced price for the product that would encourage volume sales. Then the company promoted special discounts for volume packages. Of course, the resulting question was how would the discount price affect the bottom line. What would be the total effect on unit volume, sales dollars, and gross profit? APS makes it easy to monitor these changes, and thus provides control to management.

In more general terms APS was also designed for requirements such as:

- The evaluation of alternate business strategies
- Internal analysis of corporate operations to determine product mix, manufacture *vs* purchase, lease *vs* buy, and similar policy decisions
- Investment and cash flow analysis and projection
- Project evaluation and risk analysis
- Financial planning and budgetary control in constantly changing environments.



Ken Bell Lowell Freidman
Robert Campbell Gerald Wisz Marguerite Boisvert Janet Cramer

Atlanta

Ken Bell is the new branch manager in Atlanta. He holds Master's degrees in communication from the University of Northern Colorado and in international relations from Boston University. Before joining I.P. Sharp Associates, he worked as a Russian linguist in Berlin.

Prior to his appointment as branch manager, Ken was a marketing representative in the Dallas office.

Houston

Lowell Freidman has been appointed manager of the Houston office. He graduated from Syracuse University with Bachelor of Science degrees in industrial engineering and in operations research and with an M.B.A. in finance.

Lowell previously worked as a financial consultant and programmer with Time Sharing Resources, and as an APL consultant in I.P. Sharp's Houston office. He then left I.P. Sharp to assume the position of manager of courier services at Northern Messenger and returned to I.P. Sharp in 1981.

Rochester

Robert Campbell joined I.P. Sharp Associates as marketing manager in Rochester. He graduated from Cornell University with a B.S. in marketing economics. He has considerable marketing experience garnered from his twelve-year association with the Eastman Kodak Company where he started in direct sales in New York City in both the financial and commercial markets.

In his new position with I.P. Sharp, Bob is involved with marketing new data products for customers needing corporate financial data (Form 10-K), and securities data from major and regional exchanges.

Gerald Wisz is the senior systems consultant for decision support and financial services for the Rochester region. He holds a Ph.D. from the Johns Hopkins University in operations research and econometrics.

As a captain stationed in the Pentagon, he applied military operations research techniques. He then headed up a new operations research department for Marine Midland Banks, Inc. While with Marine Midland, he held various management positions and became a vice-president.

Since joining I.P. Sharp, Jerry's main emphasis has been in the banking and financial sectors. His responsibilities to these sectors include introducing our products and services, consulting, participating in project teams, and giving seminars.

Book Ends

Grant Clarke, Toronto

Washington

Janet Cramer is the new branch manager in Washington, D.C. She has a B.A. in mathematics and physics from Smith College and an M.Sc. in operations research from Columbia University.

Janet first joined I.P. Sharp in 1972, in Toronto, as a computer operator, and then APL programmer. She returned to I.P. Sharp for one summer to develop an inventory system for Massey-Ferguson. Not able to stay away from the company, she returned in 1981 to work in Washington on an interactive computer-aided crew scheduling system, as well as numerous small systems using public data bases.

During her hiatus from I.P. Sharp, she was with Bell Laboratories in the data network analysis department.

Wayne

Marguerite Boisvert is the manager of the newly established branch in Wayne, New Jersey. She holds a B.S. in chemistry from the University of Massachusetts and a Ph.D. in chemistry from Syracuse University.

Before joining I.P. Sharp, Marguerite was a programmer/analyst in the technical systems group at American Can Company and at Syracuse University Academic Computing Center. While at Syracuse, she was also an adjunct professor in the School of Management.

New publications from the **Applications Software** group:

XTABS User's Guide *(new)*

XTABS is a software product designed to accommodate researchers in interpreting survey data. See the article "Crosstabulations with XTABS" in this issue.
October 1982, 100 pp., $7.00

There are also several new or revised software package brochures free of charge. The **Applications Software** brochure describes the wide and ever-growing range of applications software products available on the I.P. Sharp system. Other recent brochures include: **EASY—Econometric Analysis System**; **Lease Evaluation System**; **SUPERPLOT**; and the **Human Resources System**.

New **Data Bases** publications:

Sixth Annual Aviation Conference Proceedings *(new)*

In May 1982, I.P. Sharp Associates hosted its sixth annual Aviation Conference in New York City. Twelve papers were presented; these have been bound and will be available for a limited time.
August 1982, 120 pp., $5.00

Currency Exchange Rates Data Base Manual *(revised)*

The Currency Exchange Rates (CURRENCY) data base is a collection of daily exchange rates for major world currencies on various foreign exchange markets.
July 1982, 24 pp., $3.00

Australian Stock Exchange Indices Data Base Manual *(new)*

The AUSTOCK data base contains daily price and accumulation indices of the Australian Stock Exchanges.
July 1982, 18 pp., $3.00

OECD Data Base Manual *(new)*

The Organization for Economic Co-operation and Development, based in Paris, is a valuable source of international business and industrial information. The OECD data base contains statistics on major economic indicators for 25 individual OECD countries.
July 1982, 57 pp., $5.00

NPA Data Base Manual *(new)*

The National Planning Association (NPADEMOG, NPAECO) data base contains economic and demographic information about the United States, and covers more than 3 600 separate geographic areas.
September 1982, 63 pp., $6.00

Canadian Bonds Data Base Manual *(new)*

The Canadian Bonds (CDNBOND) data base comprises weekly price and yield statistics for approximately 1 150 Canadian bonds, grouped by type of issuer. Information for each bond includes bid, ask, yield, maturity date, coupon, and any privileges available.
September 1982, 36 pp., $4.00

New Dial Access Numbers

Financial and Economic Data Bases Reference Guide *(new)*

I.P. Sharp Associates provides more than eighty public data bases to our customers. Thirty or so can be classed as *financial*, and another twenty or so as *economic*. These have now been compiled in a handy pocket-sized reference guide. For each data base, the reference guide tells you what the data base is, its frequency (daily, weekly, monthly), and its timeframe (i.e. for what years data is available). The guide also shows you, for each data base, how to select data items and produce reports.
September 1982, 128 pp., $3.00

Cincinnati

30 cps 513 421-7654

Copenhagen

120 cps 01 118239

Helsinki

120 cps 90 6948144

Pompano Beach

30 cps 305 941-2501

Spokane

120 cps 509 747-0527

Warrington

120 cps 0925 54376

Winnipeg

120 cps 204 957-1562

MAILING REQUEST

- ☐ Please amend my mailing address as indicated.
- ☐ Please add the following name(s) to your Newsletter mailing list.
- ☐ Please send me a Publications Order Form.
- ☐ Please add my name to the Aviation Newsletter mailing list.
- ☐ Please add my name to the Energy Newsletter mailing list.
- ☐ Please add my name to the Financial and Economic Newsletter mailing list.
- ☐ Please add my name to the Promis Newsletter mailing list.
- ☐ Please send me information about your courses in

(City) ____________________

Name: ____________________

Title: ____________________

Company: ____________________

Address: ____________________

____

____

____

Telephone: ____________________

The Newsletter is a regular publication of I.P. Sharp Associates. Contributions and comments are welcome and should be addressed to: I.P. Sharp Newsletter, Suite 1900, 2 First Canadian Place, Toronto, Canada M5X 1E3.
Irene Shimoda, *Editor*
Mary Kopfensteiner, *Circulation*
Printed in Canada
November 1982
ISSN 0226 854X

I.P. Sharp Associates International Headquarters
Suite 1900, 2 First Canadian Place, Toronto, Canada M5X 1E3 (416) 364-5361

International Offices

Aberdeen
I.P. Sharp Associates Limited
5 Bon Accord Crescent
Aberdeen AB1 2DH
Scotland
(0224) 25298

Amsterdam
InterSystems BV
Kabelweg 47
1014 BA Amsterdam
The Netherlands
(020) 86 80 11
Telex: 18795 ITS NL

Atlanta
I.P. Sharp Associates, Inc.
1210 S. Omni International
Atlanta, Georgia 30335
(404) 586-9600

Boston
I.P. Sharp Associates, Inc.
1 Liberty Square
Boston, Massachusetts 02109
(617) 542-2313

Brussels
I.P. Sharp Europe SA
Boulevard de la Cambre 36, bte 5
1050 Bruxelles
Belgium
(02) 649 99 77

Calgary
I.P. Sharp Associates Limited
Suite 550, Bow Valley Square 4
250-6th Ave. South West
Calgary, Alberta T2P 3H7
(403) 265-7730

Canberra
I.P. Sharp Associates Pty. Ltd.
16 National Circuit
Barton, A.C.T. 2600
Australia
(062) 73-3700

Chicago
I.P. Sharp Associates, Inc.
Suite 3860
55 West Monroe Avenue
Chicago, Illinois 60603
(312) 782-3177

Copenhagen
I.P. Sharp ApS
Østergade 24B
1100 Copenhagen K
Denmark
(01) 11 24 34

Coventry
I.P. Sharp Associates Limited
7th Floor B Block
Coventry Point, Market Way
Coventry CV1 1EA England
(0203) 56562

Dallas
I.P. Sharp Associates, Inc.
Suite 1148, Campbell Center
8350 Northcentral Expressway
Dallas, Texas 75206
(214) 369-1131

Denver
I.P. Sharp Associates, Inc.
Suite 416
5680 South Syracuse Circle
Englewood, Colorado 80111
(303) 741-4404

Dublin
I.P. Sharp Associates Limited
Segrave House
Earlsfort Terrace
Dublin 2, Ireland
(01) 763605

Düsseldorf
I.P. Sharp GmbH
Kaiserswertherstrasse 115
4000 Düsseldorf 30
West Germany
(0211) 45 20 52

Edmonton
I.P. Sharp Associates Limited
2358 Principal Plaza
10303 Jasper Avenue
Edmonton, Alberta T5J 3N6
(403) 428-6744

Halifax
I.P. Sharp Associates Limited
Suite 706, Cogswell Tower
2000 Barrington Street
Halifax, Nova Scotia
B3J 3K1
(902) 423-6251

Helsinki
TMT-Team OY (Agent)
Kalevankatu 33 A
P.O. Box 452
SF-00101 Helsinki 10, Finland
(9) 0-6946344

Hong Kong
I.P. Sharp Associates (HK) Limited
Suite 606, Tower 1
Admiralty Centre, Hong Kong
5-294341

Houston
I.P. Sharp Associates, Inc.
Suite 375, One Corporate Square
2600 Southwest Freeway
Houston, Texas 77098
(713) 526-5275

London, Canada
I.P. Sharp Associates Limited
Suite 510, 220 Dundas Street
London, Ontario N6A 1H3
(519) 434-2426

London, England
(European Headquarters)
I.P. Sharp Associates Limited
132 Buckingham Palace Road
London SW1W 9SA, England
(01) 730-4587
Telex: 8954178 SHARP G

Los Angeles
I.P. Sharp Associates, Inc.
Suite 1230
1801 Century Park East
Los Angeles, Ca. 90067
(213) 277-3878

Madrid
I.P. Sharp Associates Limited
Serrano 23, Piso 8
Madrid 1, Spain
(91) 276 70 54

Melbourne
I.P. Sharp Associates Pty. Ltd.
520 Collins St., 7th Floor
Melbourne, Victoria
3000, Australia
(03) 614-1766

Mexico City
Teleinformatica de Mexico S.A.
(Agent) Mail to:
Arenal N 40, Chimalistac
Mexico 20 D.F., Mexico
(905) 550-8033

Miami
I.P. Sharp Associates, Inc.
Suite 240
15327 N.W. 60th Avenue
Miami Lakes, Florida 33014
(305) 556-0577

Milan
Informatical Society Italia Srl
(Agent)
Via Eustachi 11
20129 Milan, Italy
(02) 221612

Montreal
I.P. Sharp Associates Limited
Suite 1610
555 Dorchester Boulevard W.
Montreal, Quebec H2Z 1B1
(514) 866-4981

New York
I.P. Sharp Associates, Inc.
Suite 210
230 Park Avenue
New York, N.Y. 10169
(212) 557-7900

Newport Beach
I.P. Sharp Associates, Inc.
Suite 1135
610 Newport Center Drive
Newport Beach, Ca. 92660
(714) 644-5112

Oslo
I.P. Sharp A/S
Postboks 486 Sentrum
Dronningens gate 34
Oslo 1, Norway
(02) 41 17 04

Ottawa
I.P. Sharp Associates Limited
Suite 600, 265 Carling Ave.
Ottawa, Ontario K1S 2E1
(613) 236-9942

Palo Alto
I.P. Sharp Associates, Inc.
Suite 201, 220 California Ave.
Palo Alto, Ca. 94306
(415) 327-1700

Paris
I.P. Sharp SARL
9 Rue de Cirque
75008 Paris-la-Défense 92086
France
(1) 225 9820

Philadelphia
I.P. Sharp Associates, Inc.
Suite 604, 437 Chestnut St.
Philadelphia, Pa. 19106
(215) 925-8010

Phoenix
I.P. Sharp Associates, Inc.
Suite 503
3033 N. Central Avenue
Phoenix, Arizona 85012
(602) 264-6819

Rochester
(United States Headquarters)
I.P. Sharp Associates, Inc.
1200 First Federal Plaza
Rochester, N.Y. 14614
(716) 546-7270
Telex: 646318

Rome
Informatical Society Italia Srl
(Agent)
Piazza Della Rotonda 2
00100 Rome, Italy
(06) 656-5925

San Francisco
I.P. Sharp Associates, Inc.
Suite C-415, 900 North Point St.
San Francisco, Ca. 94109
(415) 673-4930

San Jose
I.P. Sharp Associates, Inc.
Suite 212, 230 California Ave.
Palo Alto, Ca. 94306
(415) 327-1725

Saskatoon
I.P. Sharp Associates Limited
Suite 303, Financial Bldg.
230-22nd St. E., Saskatoon
Saskatchewan S7K 0E9
(306) 664-4480

Seattle
I.P. Sharp Associates, Inc.
Suite 223, Executive Plaza East
12835 Bellevue Redmond Road
Bellevue, Washington 98005
(206) 453-1661

Seoul
Daewoo Corporation (Agent)
541, 5-GA, Namdaemoon-Ro
Jung-G (CPO Box 2810)
8269 Seoul, Korea.
771-91/2
Telex: DAEWOO K23441-5

Singapore (Far East H.Q.)
I.P. Sharp Associates(S) Pte. Ltd.
Suite 1003, PIL Building
140 Cecil Street
Singapore 0106
Republic of Singapore
2210494
Telex: RS 20597 IPSAS

Singapore
Singapore International
Software Services Pte. Ltd.
Suite 1501, CPF Building
79 Robinson Rd.
Singapore 0106
Republic of Singapore
2230211

Stockholm
I.P. Sharp AB
Kungsgatan 65
S111 22 Stockholm, Sweden
(08) 21 10 19

Sydney (Australian H.Q.)
I.P. Sharp Associates Pty. Ltd.
8th Floor, Carlton Centre
55 Elizabeth St.
Sydney, New South Wales 2000
Australia
(02) 232-6366

Tokyo
INTEC, Inc. (Agent)
37-18, 3-Chome, Hatagaya
Shibuya-ku, Tokyo 151
Japan
(03) 320-2020

Toronto
(International Headquarters)
I.P. Sharp Associates Limited
Suite 1900
2 First Canadian Place
Toronto, Ontario
M5X 1E3
(416) 364-5361
Telex: 0622259

Vancouver
I.P. Sharp Associates Limited
Suite 902, 700 West Pender St.
Vancouver, B.C. V6C 1G8
(604) 687-8991

Victoria
I.P. Sharp Associates Limited
Chancery Court
1218 Langley Street
Victoria, B.C. V8W 1W2
(604) 388-6365

Vienna
I.P. Sharp Ges.mbH
Renngasse 4
A-1010 Wien, Austria
(0222) 66 42 48

Warrington
I.P. Sharp Associates Limited
1-3 Dolmans Line
Warrington, Cheshire
WA1 2ED England
(0925) 50413/4

Washington
I.P. Sharp Associates, Inc.
Suite 305, 2033 K Street N.W.
Washington, D.C. 20006
(202) 293-2915

Wayne
I.P. Sharp Associates, Inc.
Suite 301
155 Willowbrook Road
Wayne, New Jersey 07470
(201) 785-8050

White Plains
I.P. Sharp Associates, Inc.
Suite 312 West
701 Westchester Avenue
White Plains, New York 10604
(914) 328-8520

Winnipeg
I.P. Sharp Associates Limited
Suite 208
213 Notre Dame Avenue
Winnipeg, Manitoba R3B 1N3
(204) 947-1241

Zurich
I.P. Sharp AG
Fortunagasse 15
8001 Zurich, Switzerland
(01) 211 84 24

SHARP APL Communications Network

APL OPERATOR VOICE (416) 363-2051 COMMUNICATIONS (416) 363-1832

Our private, packet-switched network connects with the Value Added Networks – Datapac, Datex-P, PSS, Telenet, and Tymnet – to provide access from the 35 countries listed below:

Argentina • Australia • Austria • Bahrain • Belgium • Bermuda • Canada • Chile • Denmark • Dominican Republic • England • Finland • France • Germany • Hong Kong • Ireland • Israel • Italy • Japan • Luxembourg • Mexico • The Netherlands • New Zealand • Norway • The Philippines • Portugal • Puerto Rico• Scotland • Singapore • Spain • Sweden • Switzerland • Taiwan • United Arab Emirates • U.S.A.

SHARP APL is accessible from over 500 places via a local telephone call. Please ask your nearest I.P. Sharp office or representative for a complete list of access points and access procedures. Our private network also connects with the worldwide Telex network via the Rochester, New York and Amsterdam nodes.