the I.P. Sharp newsletter

DECEMBER 1977
Volume 5 Number 6

GRAPHICS UPDATE

Dave King

Computer graphics has two main applications in the business world, namely diagrams or illustrations, and graphs. Graphics development over the past few months has been aimed at these areas and two new workspaces have been released:

- 3 *GREDIT* - a graphics editor, and
- 3 *GRAFPLOT* - a general graph plotting package.

GRAPHICS PLOT: Graphs provide a convenient means of summarizing large quantities of data. The workspaces 3 *PLOT* and the now defunct 3 *LPLOT* were limited by the fact that they were locked, monolithic functions. Thus, if they did not meet the user's needs, there was little to do but live with the limitations, or write a specialized package. In order to improve the quality of the plots, three new plot types were added to 3 *PLOT* in the spring of this year (*LINEPLOT*, *POINTPLOT* and *LINENORM*). The quality was improved, but to get around the limitations of locked functions, 3 *GRAFPLOT* was developed. It provides functions to scale and plot the data, label and decorate the graph, draw axes, produce a legend, and much more.

Several plot types are available (normal point to point plots, histograms, stock market type plots) which may also appear together on one graph. Each plot can be drawn with a different flavour of dashed lines, hatched above or below, drawn right side up, upside down, or sideways. In other words, the user controls GRAFPLOT and **not** vice versa. If a particualr plot type is not currently supported, it is easily added.

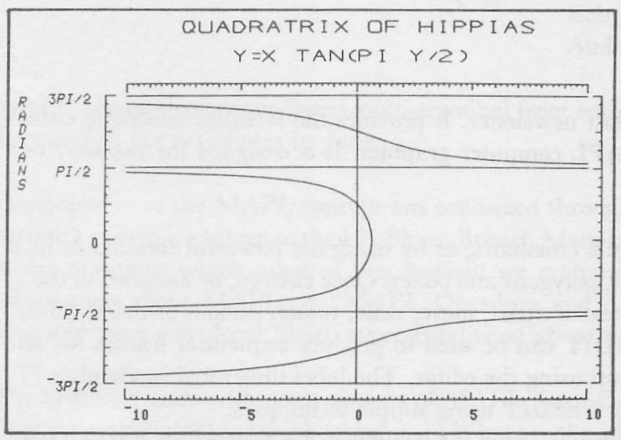


Figure A: Normal point to point plot

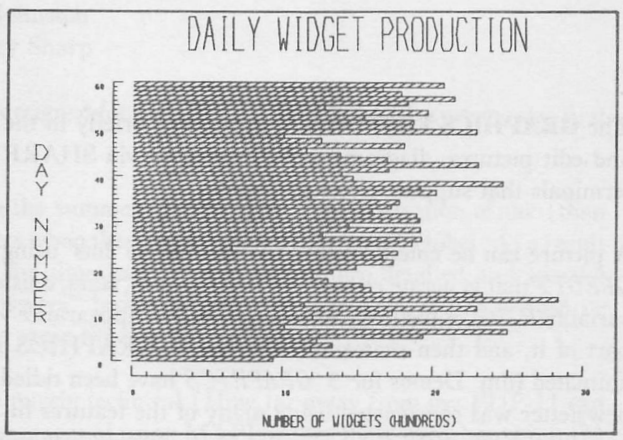


Figure B: Sideways histogram with cross-hatching

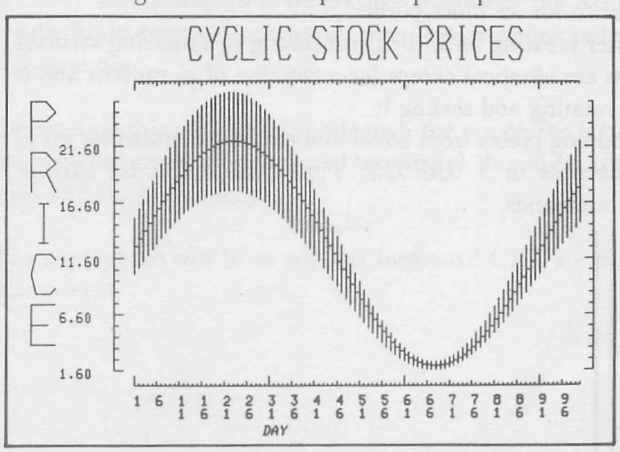


Figure C: Stock Market plot

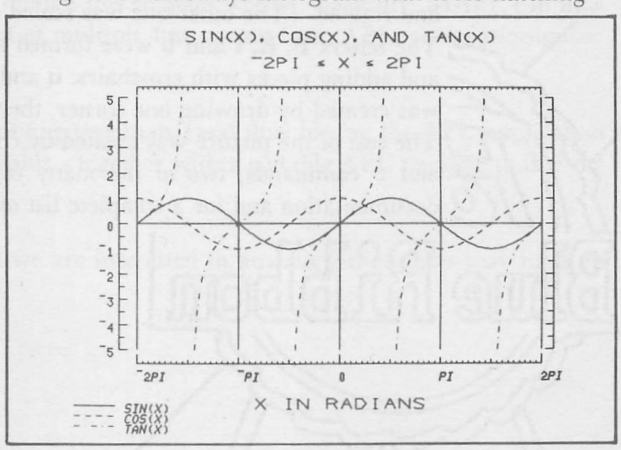


Figure D: Point to point plot with dashed lines and reference lines

GRAPHICS (continued)

A 'Compiler' was written to aid conversion from 3 *PLOT* to 3 *GRAFPLOT*: copy| *CO*| *CHARS* and your data from the 3 *PLOT* workspace you are using, and execute the function *COMPILE*. You now have a monadic function *PLOT* which will produce a graph very similar to the original 3 *PLOT* version. *PLOT* is composed only of *GRAFPLOT* functions. Complete on-line documentation is accessible via *DESCRIBE*, and there is a demo. The example below is a graph originally drawn with 3 *PLOT* and then compiled.

Figure E: 3 *PLOT* version

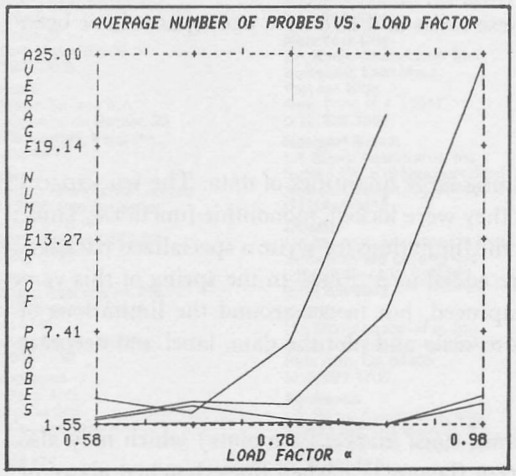
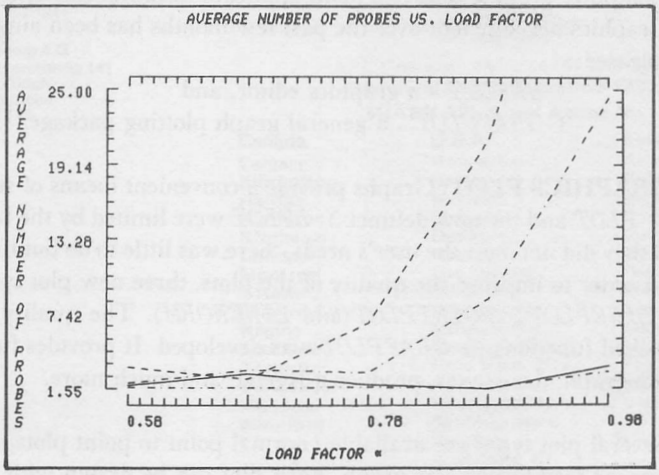


Figure F: Unmodified compiled version



Acknowledgement: Many thanks to Leslie Goldsmith, the author of 3 *PLOT*, for these graphs.

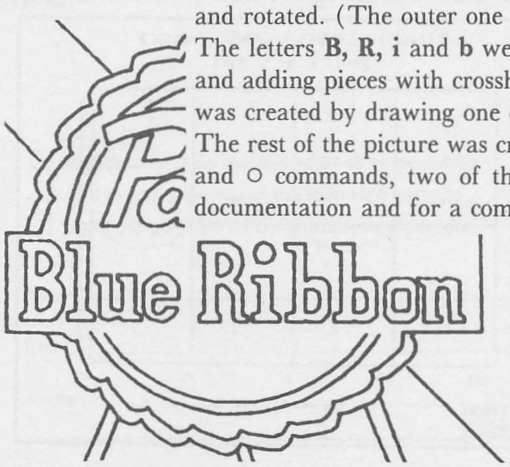
The **GRAPHICS EDITOR** was introduced briefly in the last newsletter. It provides the facilities needed to enter and edit pictures, diagrams and illustrations via SHARP APL computer graphics. It is designed for use only on terminals that support crosshair input.

A picture can be entered either by 'joining the dots' using the crosshairs, or by using the powerful commands in 3 *GREDIT* that generate circles and circular arcs, stars, daisies, polygons and boxes. Once entered, or assigned to the variable *RINP*, a picture can be improved in appearance - tear it apart, move, scale, rotate, smooth or delete all or part of it, and then reassemble the pieces. GRAPHICS EDIT can be used to produce sequential frames for an animated film. Demos for 3 *GRAPHICS* have been tidied up using the editor. The label illustrated in October 77 newsletter was constructed using many of the features in 3 *GREDIT* using simple techniques:

To make a scalloped border, one arc was drawn using the circular arc generator, then translated and rotated. (The outer one was scaled down for an inner border.)

The letters **B, R, i** and **b** were formed after creating an **l**, then translating it, removing sections and adding pieces with crosshairs. **u** and **n** are identical except for a rotation of pi radians and **o** was created by drawing one corner, then rotating and scaling it.

The rest of the picture was created by chopping pieces from boxes and circles generated by the □ and ○ commands, two of the many commands in 3 *GREDIT*. Type *DESCRIBE* for further documentation and for a complete list of commands.



APL UNDER DISCUSSION - MINNOWBROOK, NEW YORK

Lib Gibson,
I.P. Sharp Europe S.A. (Brussels)

The Minnowbrook conference was hosted by Syracuse University to continue and improve the dialogue among those active in APL. The workshop built on progress achieved at a similar conference held at Queen's University last year, and two previous Minnowbrook conferences. The discussion among participants undoubtedly presages many enhancements to APL which will actually come to pass. On the other hand, the emphasis of such a meeting is to elicit current thinking of the participants before commitments to implementation are made. Some of the presentations were supported by written material, and these will eventually be published, as well as reports in APL Quote Quad (see page 7) on the various sessions.

Workspace Interchange

After discussions at Queen's, three workspace interchange mechanisms had been implemented: by Bob Bernecky (Sharp, SATN-22), Mike David (STSC) and Dana Cartwright (Syracuse University). At the Minnowbrook Workspace Interchange Session, there was overwhelming support for The Bernecky Approach. An ad hoc committee appointed by STAPL, consisting of representatives of virtually all manufacturers with APL implementations, was able to reach an agreement before the end of the conference on a proposed standard which is an extension of Bernecky's. A draft standard will appear in the December issue of APL Quote Quad soliciting comments from the APL community in general, with the final standard, endorsed by STAPL, published in the spring Quote Quad.

Commonality

Although APL is distinguished by remarkable consistency among various implementations of the language, a preliminary survey carried out by Clark Wiedmann (University of Massachusetts) indicates there are still areas of divergence in APL systems. Order of execution, although closer to compatibility than indicated in a previous survey, still shows differences in treatment of expressions known as soft-core pornography - for example, responses to the expression

```
A←2
(A←10)×A
```

were variously `20` or `100` . Such SHARP APL features as diamond, end-of-line comments, and the ability to allow a dyadic user-defined function to be used monadically, are not widely available. But the major differences lay in the area of file capability, where systems differed so significantly as to make it difficult to compose a set of meaningful questions relevant to all.

General Arrays

The topic of arrays of arrays (or general arrays) aroused great interest, not to mention controversy. In brief, a general array can be thought of as something that has shape, and whose contents are items that can themselves be arrays. APL arrays also have shape, but their contents must be elementary objects of APL (either characters or numbers), not other arrays. Two different approaches to the problem were discussed in detail (More/Brown/Ghandour-Mezei, and Gull-Jenkins, which use slightly different definitions). The encouraging thing about the different approaches was the degree of agreement between them, with only the edge cases differing. Moreover, Jim Brown (IBM) reported on simulation work by Don Orth using general arrays in real applications, where so far there has been no practical requirement to adopt either of the definitions.

An interesting paper in a session on extensible APL by Louis Robichaud, described how Laval University has used existing facilities to provide an ability to handle general arrays.

(continued)

MINNOWBROOK (continued)

Complex Numbers

A recommendation based on an elaboration of one of the alternatives given by Penfield in a recent note published in APL Quote Quad (Volume 8 No.1), emerged from a session devoted to the subject of complex numbers. Four new primitives (real part, imaginary part, phase and magnitude) to be defined on complex numbers would be implemented through use of new arguments for the circle functions. The proposed standard will be published in the next Quote Quad, with a final standard promised in June.

New Scope Rules

A proposal was made to extend the types of scope of variables beyond just local and global. This would include provision of variables which cannot be seen either by their calling or their called functions, for instance. Five types of scope seem useful and the proposed facility would enable a user to declare all unnamed variables to default to certain scope type.

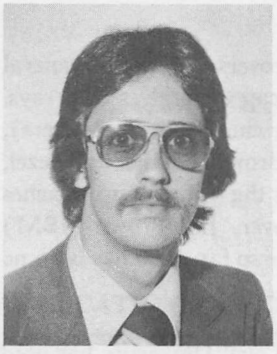
Packages

A package is a package is a package - except when it's someone else's package. Ira Greenberg (U. Mass.) presented a proposal for a package concept to improve APL security. Jim Ryan (Burroughs), working independently, proposed a concept of namespace, an extension of Greenberg's package concept. The essence of these proposals would be the provision of a capability to seal or isolate a group of objects in a 'package' or 'namespace' so that only objects deliberately named can communicate with the external environment. 'Exports' from such a package could be seen or used by users while the package could only use variables external to it if they were declared as 'imports'.

The Minnowbrook Workshop brought together a diverse, enthusiatic group of people, representing several competing vendors and manufacturers, working together in a spirit of openness and cooperation. Syracuse University, and particularly Garth Foster, are to be commended for their initiative in hosting the meeting.

SHARP NEWS

NEW YORK -



Keith Iverson

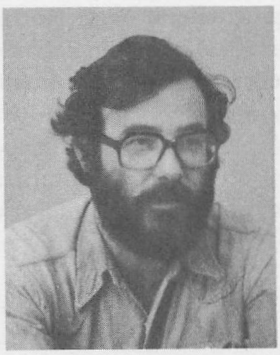
Keith Iverson, Branch Manager of our New York City office (in the Pan Am Building, Manhatten), joined Sharp over seven years ago to work in operations. After six months he became a contract programmer. He helped to implement the original Financial Post data base and spent a year in London when that office first opened. During the last three years Keith worked on a Logistics project at Massey Ferguson. With him in the New York office are **David Osekavage** and **Gary Brown**.

LONDON -

Fred Perkins is about to take over as MANAGING DIRECTOR of the U.K. Company and **John Bassingthwaighte** will be returning to Vancouver in February. The next Newsletter will have appropriate photographs and a fuller write-up of this important event.

SHARP NEWS (continued)

TORONTO - **Richard Lathwell** joined the Toronto APL System Design Group (the Zoo) in October. His specialties are the fundamentals of APL, the design of APL systems, and the use of APL in system design.



Richard Lathwell

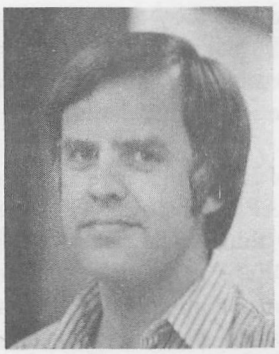
Born and raised in Calgary, he attended the University of Alberta (B.Sc Mech. Eng. 1965) where, as a 'U. of A. computer bum', he learned APL (then called Iverson's notation), and met K.E. Iverson in 1963.

He joined Iverson's group at the IBM Thomas J. Watson Research Center in 1966 and together with L.M. Breed and R.D. Moore implemented APL 360. In 1969, Iverson's group left IBM Research to form the IBM Philadelphia Scientific Center under Adin Falkoff, where Richard was Manager of APL Implementation and System Design from 1969 to 1974. He designed the APL Shared Variable Processor, and built a prototype APLSV system in 1971. This system subsequently received wide acceptance within IBM, and was announced as a product in 1973.

In 1974, the Philadelphia Scientific Center was closed, and the IBM APL Design Group, consisting of Adin Falkoff, Ken Iverson and Richard Lathwell, was appointed. As a member of this group, he was heavily involved with APL within IBM, including the 5100 program and internal uses of APLSV.

PALO ALTO - **Joey Tuttle** joined I.P. Sharp Associates and will be working out of Palo Alto, California, where he lives.

"The main reason for my joining Sharp is my continuing interest in APL, which I see as the most reasonable way for people to use computers. After using FORTRAN to implement a model for some graduate work in atmospheric physics, I decided that I liked computing equipment, but never wanted to be a 'programmer'. Rejoining IBM in 1968 I continued my interests in electronics by designing and constructing various instrumentation systems used to develop and test magnetic data storage products. Shortly after returning to IBM I discovered APL. Since then I have done a good deal of programming, almost exclusively in APL (not an unusual story).

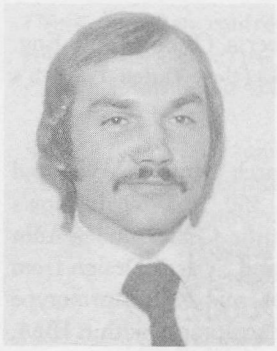


Joey Tuttle

The underlying purpose of instrumentation is to provide data to support decisions, and early on I became interested in data collection and analysis using APL..Wanting to make various alien data (e.g. operating system files) available to an APL user led me to become involved in the communication aspects of the APL environment. I joined the APL Design Group at IBM's Philadelphia Scientific Center in 1971. My work since then has been primarily related to defining the APL environment.

Much of my spare time is taken up by family activities with my wife and three daughters and I enjoy bicycling, reading and listening to music. I also enjoy talking to people and have an implicit interest in education. I'm looking forward to working with the people at I.P. Sharp Associates, and helping to spread the use of APL to a larger and larger audience."

LETTER FROM LONDON Valerie Lusmore



Alan Harrison, BNF

The number of people traditionally involved with solving a management problem with the help of computers is a daisy chain: manager→ analyst→ programmer→ key-punch operator→ computer operator→ ..... Each time another link is added, the time needed to solve the problem goes up exponentially, as a lot of time is necessary for communication along the chain.

APL reduces the number of people involved to one or two - the person with the problem, and the person solving it. Systems can start small and grow in response to patterns of use. Notably, file structure can be changed to remain efficient as the system changes - all data is accessed through basic read/write functions so that only these functions have to be changed when the file structure changes.

A good example of technique development is the Windscale Management Information System, INFO, developed by Alan Harrison of **British Nuclear Fuels**. Situated very near the Lake District in the North of England, BNF was one of the early British users of SHARP APL . Alan had done quite a lot of scientific problem-solving with APL but, before INFO, had not tackled a 'commercial' problem this way. In early 1976, a requirement for a Management Information System arose. The data was already available in machine-readable form from BNF's own computers so all they needed was a system. APL was chosen for the design because the system was required for a new department of BNF. Not knowing exactly how the data would be used, the design had to be extremely flexible, easily modified and extensible. Design and implementation time had to be minimal as the system was required quickly and interactive use was highly desirable.

A design feature of INFO that is relatively unusual - and which has proved extremely successful - is that the user, not familiar with APL, constructs APL statements in answer to the prompts. Thus instead of developing a syntax analyser (a costly thing to do), the APL interpreter is utilised. Simple recovery instructions allow re-entry for SYNTAX ERRORs. For example, consider the following question to a typical warehouse/stock-control system. 'How many packs containing between 10 and 100 bottles of aspirin have we in stock?'. Using INFO the search criteria along the stock-control data base might look like:

```
INPUT: CODE=1013 (1013 is the code for aspirin)
INPUT: PACKSIZE>10
INPUT: PACKSIZE<100
```

and so on. The constraints typed by the user are actual APL statements, the keywords *CODE* and *PACKSIZE* being APL functions for this data-set.

The original basic system fulfilled the design criteria well. A working system was produced in two days, the first limited data base handed over to users within a week, and full implementation was within 6 weeks. Extensions occurred during that implementation to make INFO a general information system with the introduction of particular data-sets containing specialised information, for example stock control, payroll data, etc. Users are authorised to use particular data-sets only and all access is controlled through one 'master' INFO workspace. The users (Alan refers to them as customers), are from about 5 different departments and typically, are not experienced computer users. However, they quickly became adept at forming constraints to retrieve particular data. Four actual modes of use are available to them. These are (a) selection of data-set, (b) definition of constraints, as many as required, (c) retrieval of the data and (d) further manipulations of the data, either through general pre-programmed functions (for example - TOTAL) or through their own APL routines or using the terminal in desk-calculator mode.

Technical Supplement – 13

UTILITIES - PART TWO

Peter Teeson

One of the more frequent problems we run into is the manipulation of character arrays. For example, significant space savings can accrue if we store a name and address list as a series of character vectors with embedded carriage returns rather than as character matrices.

However, to edit the name and address it is usually simpler to deal with a matrix.

We may also wish to left or right justify, or trim any excess blank columns, before 'vectorizing' the result.

To help us we can use the following utilities:

```
VTM  ⍝ VECTOR TO MATRIX
MTV  ⍝ MATRIX TO VECTOR
RJ   ⍝ RIGHT JUSTIFY ARRAY
LJ   ⍝ LEFT JUSTIFY ARRAY
DEC  ⍝ 'DE-COMMENT' ARRAY
TRM  ⍝ TRIM EXCESS BLANK 'COLUMNS'
```

VTM has been around for quite some time under the alias of *AJR* . It converts a vector right argument into a matrix, using the left argument vector of delimiters.

Comments from an origin 1 version follow, and the program itself can be found in component 3 of file 1142021 *UTILS* . The original one-liner version is in component 4. (Note 1)

```
R←DL VTM VEC;V
⍝ VECTOR TO MATRIX USING DELIMITERS DL.
⍝ ESTABLISH DEFAULT <DL> IF ELIDED.  RAVEL BOTH ARGUMENTS.
⍝ MASK TO COMPRESS OUT DL'S
⍝ MASK TO EXPAND TO WIDEST ROW
⍝ MAKE THE MATRIX
```

Readers are invited to submit a better solution, together with an explanation of why it is better. (I know of one such that reduces *WSFULL* chances).

What does the *MTV* function look like? Please append your solutions, comments, etc. to file 1142021 *UTILS* or mailbox *PHT*.

The following *LJ* function uses scan in a pleasant way.

```
    ∇R←LJ A
[1]   R←(+/∧\' '=A)⌽A  ⍝ LEFT JUSTIFY ARRAY
    ∇
```

What does *RJ* look like?

UTILITIES (continued)

Another useful function is *DEC* or *DECOMMENT*, for want of a better name. That is, given some character array extract up to *DL* from each 'row'. An example would be to remove comments from the matrix form of a program.

```
R←DL DEC A
  ⍝ ESTABLISH DEFAULT DL IF ELIDED
  ⍝ EXTRACT UP TO DL FROM EACH 'ROW' OF A.
```

The function can be found in component 7 of file 1142021 *UTILS*.

Finally, it is often useful to trim an array, that is to drop off leading and trailing blank columns (Note 2).

The problem is fairly trivial if we are dealing with matrices, but what do we do with arrays in general?

The following surprising function was written by Doug Forkes (*DLF*) - Note 3.

```
    ∇R←TRM R;RR
[1]   RR←⍴R
[2]   R←((×/¯1↓RR),×/¯1↑1,RR)⍴R
[3]   R←⌽(∨≠∨\~R∈' ')/R
[4]   R←⌽(∨≠∨\~R∈' ')/R
[5]   R←((¯1↓RR),¯1↑⍴R)⍴R
[6]   ⍝ SCALARS BECOME VECTORS.
    ∇
```

As in the last issue, I solicit your response on the definition of what a utility is (or should be), as well as your versions of, or improvements on, the functions offered. Many thanks to those of you who did respond.

Please have your say, either by mailbox (*PHT*) or by appending to file 1142021 *UTILS*.

Note 1 Much could, and has been, said about one-liners. The purpose here is to try to emphasize the basic parts of the algorithm.

Note 2 Of course this means do not remove any of the 'middle' blank columns.

Note 3 *AJR* is courtesy of Al Rose of STSC (who gave it to Bob Bernecky of I.P. Sharp Associates, many years ago). Doug Forkes (Sharp) came up with the *TRM* function when we were discussing the problems of the general case.

HOW TO TRAIN AN NTASK (CHEAPLY)

Steve Kohalmi

NTASKs allow users to perform jobs at a lower cost and can add to the security of a system. Since CPU charges are lower for NTASKS and since they are executed immediately, they are an attractive alternative to TTASKs. Security is increased because a user is one step removed from sensitive code within a system and so has less opportunity to experiment and perhaps cause damage.

The technique was initially developed for the MAPL system. Since MAPL utilizes one APL line for multiple terminals, it was desirable to:

1. Allow the TTASK to act as an I/O handler, and make minimum use of the communication line by avoiding heavy processing;

2. Separate the interface code in the TTASK from the APL applications software of the NTASK;

3. Enhance restartability
   (A line drop will not stop the NTASK at an awkward point in an update - possibly complicated recovery procedures occur only in cases of system failure, much less frequently);

4. Run multiple NTASKS by means of an extension of this scheme, which allows
   - specialization of the duties of each NTASK,
   - a capability of execution priority,
   - independence of the operations of each NTASK.

Consider the following programs:
The function *TDRIVER* is the TTASK master of an NTASK, where the TTASK controls queueing of input commands and the printing of output, while the actual process is carried out independently by an NTASK. That is, *NDRIVER* is kept in a 'wait state' when there is no processing to be done. *TDRIVER* allows the NTASK to continue as soon as some command is in the input queue.

TTASK

```
    ∇TDRIVER;INQ;OUTQ;I
[1]  ⎕IO:INQ←ΔSTIE '1234567 INFILE'
[2]   OUTQ←ΔSTIE '1234567 OUTFILE'  ⍝ ASSUME INQ AND OUTQ ARE TIE NUMBERS
[3]   ⎕HOLD INQ
[4]   STARTNTASK
[5]  L1:→('→'=1↑IN←⍞)⍴ENDRTN  ⍝ENTER COMMAND OR → EXIT
[6]   IN ⎕APPEND INQ  ⍝ APPEND THE COMMAND TO THE Q TO BE PROCESSED BY NTASK
[7]   ⎕HOLD INQ  ⍝ RELEASE AND REHOLD FILE I.E. LET NTASK GO
[8]   →(=/I←2↑⎕SIZE OUTQ)⍴L1  ⍝ CHECK FOR PENDING OUTPUT
[9]   ⎕←⎕READ OUTQ,1↑I  ⍝ OUTPUT
[10]  ⎕DROP OUTQ,1      ⍝ REMOVE OUTPUT
[11]  →L1
[12] ENDRTN:
    ∇
```

TRAINING AN NTASK (continued)

NTASK

```
    ∇NDRIVER;INQ;OUTQ;I
[1]  ⎕IO:INQ←ΔSTIE '1234567 INFILE'  ⍝ ASSUME INQ AND OUTQ ARE TIE NUMBERS
[2]      OUTQ←ΔSTIE '1234567 OUTFILE'
[3]  L1:→(=/I←2↑⎕SIZE INQ)⍴WAITΔSTATE
[4]  L2:PROCESS ⎕READ INQ,1↑I  ⍝ ASSUME THIS GENERATES OUTPUT TO THE OUTPUT FI
LE
                                    ⍝AND PROCESSES THE COMMAND IN THE FILE CMP.
[5]   ⎕DROP INQ,1  ⍝ DROP COMPLETED COMMAND FROM Q.
[6]   →L1
[7]  WAITΔSTATE:⎕HOLD INQ ◇ ⎕HOLD''
[8]   →(≠/I←2↑⎕SIZE INQ)⍴L2
    ∇
```

Suppose that *TDRIVER* is executed, and it in turn starts *NDRIVER* as an NTASK. The result is that the NTASK waits for commands from the TTASK. Whenever the input queue is empty the NTASK goes into a wait state (no overhead) until something is put into the input queue by *TDRIVER*. *TDRIVER* momentarily releases its hold on the input file. The NTASK, being released, now has control of the input file. Control is immediately withdrawn by ⎕*HOLD* '' (failure to do this would hold the TTASK) and the NTASK continues processing **without** ⎕*HOLD*s until the input queue is empty.

In the meantime *TDRIVER* regains its hold on the file, but this does not affect the NTASK until *NDRIVER* executes a hold on the file - when it has no work. Note that if *NDRIVER* awakens and there is no work (line [8]) then the NTASK signs off since the TTASK is no longer holding the file, that is the TTASK signed off or was in trouble.

In this scheme you have two tasks that process independently (given that there is work to do neither is forced to wait for the other) where the TTASK has an NTASK as a slave to carry out commands passed to it by files. The overhead for using this is in the appending and reading of the data being passed back and forth. When the NTASK runs out of work it goes to sleep (costing nothing while in this state). There is no overhead since there is no need for occasionally waking up to look for work as is the case when using ⎕*DL* to create a similar set-up. When the NTASK wakes up it knows that it is okay to run and it should have work to do. If the queue is empty then the TTASK has gone away (voluntarily or not!), so there is no need for the NTASK to check explicitly that all is going well.

Please contact Steve Kohalmi (mailbox *SKO*) for further information.

***CONTEST NO. 5 - W-WRE-WRECK-WRECKURSION ***

Jerry Cudeck

Consider the recurrence relations defined by:

(I) `R[T]←A[T]×R[T-1]` with boundary condition `R[⎕IO]←A[⎕IO]`

(II) `R[T]←M[T]×A[T]+R[T-1]` with boundary condition `R[⎕IO]←A[⎕IO]×M[⎕IO]`

Note:

(1) A and M are given vectors of length $N \geq 0$

(2) **No** element of M is equal to zero.

Such recurrence relations present themselves quite frequently in a considerable number of diverse application areas.

Example 1 - Accumulation of deposits at varying time-dependent interest rates.

Let

`M←1.05 1.06 1.07 1.08...`		Interest factors during time intervals 1,2,3 and 4.
`A←100 150 125 210 ...`		Amounts deposited at the start of each interval.
`R[1]←100`	...	Boundary condition (origin 1 assumed).

Then, recurrence relation (I) gives the accumulated amount of the fund immediately after each deposit is made.

Example 2 - Polynomial factoring and root extraction.

Suppose we are given the polynomial `(X*3)+(2×X*2)+(3×X)+4`
(i.e. the cubic polynomial with coefficient vector `1 2 3 4`).
Suppose further that we make the rather uninspired guess that `0.5` is a root of the above polynomial. To test this guess, we might divide the given polynomial by `X-0.5` .
The quotient of this division yields the polynomial `(X*2)+(2.5×X)+4.25` with a remainder term of `6.125`, thereby indicating that our guess was in error.

Rather than performing this division, we can again invoke recurrence relation (I), where:

`A←1 2 3 4`	...	The original coefficient vector
`M←4⍴0.5`	...	The proposed root
`R[1]←1`	...	Boundary condition (origin 1 assumed).

The result generated is readily seen to be `R←1 2.5 4.25 6.125`

That is to say, `(¯1↓R)` is the coefficient vector of the reduced polynomial and `(¯1↑R)` is the remainder term.

Y 113

****ERRATUM****

RECURRENCE RELATION (I) ABOVE
SHOULD READ:

(I) R[T]←A[T]+M[T]×R[T-1]

Example 3 - Prediction of surplus in inventory control problems.

Suppose that the results of earlier analyses indicate that over the next 6-month period, the proportions of a particular stock item that will be sold are:

```
P←0.85  0.89  0.78  0.82  0.91  0.89
```

Suppose further that these analyses indicate that in order to meet the anticipated demand for this stock item, the purchasing department will be required to place orders at the start of each month for numbers of units given by:

```
A←1000  1100  850  900  1000  1000
```

Applying recurrence relation (II), where A is as given above and where:

```
M←1-P          ...  Proportion of items remaining in stock,
R[1]←150       ...  Boundary condition (origin 1 assumed),
```

The expected number of surplus units in stock at the end of each month is given by:

```
R←150  137.5  217.25  201.105  108.09945  121.8909359
```

Armed with the foregoing examples, we formally put out a call for a dyadic function of the form:

```
R←A WRECK M
```

where A and M are as specified in Notes (1) and (2) above, and where the result R is the vector of length N containing the values that would be generated by successive applications of recurrence relation (I), paying due note to the stated boundary conditions.

Additional Requirements

(1) *WRECK* must be a loop-free, non-recursive function.

(2) If A is replaced by $A \times M$ in the left argument of *WRECK*, the result R obtained from

```
R←(A×M) WRECK M
```

is the vector of length N that would be generated by successive applications of recurrence relation (II).

Entries will be judged on completeness, speed, conciseness, minimality of intermediate storage requirements, and elegance. The deadline for submissions is March 1st,1978. Please □*APPEND* your packaged function along with your name and address to file `999 CONTEST` - or submit directly to the address below. In the event of identical winning submissions, the entry bearing the earliest □*RDCI* time stamp or post mark will get the prize.

Jerry Cudeck (mailbox code *JHC*)
I.P. Sharp Associates
145 King St. West, Toronto.
M5H 1J8 - CANADA

LETTER FROM LONDON (continued)

Alan feels that the success of the system was mainly due to its evolutionary approach. This technique offers advantages to both designer and user as they work together towards the solution. The designer can produce the basic tool very quickly and this can then evolve towards an optimum solution, especially in terms of cost. Meanwhile the user has something on which to cut his teeth while refining his definition of the requirements of the system. The problems of 'That's not in the spec,' do not arise, as the specifications evolve out of use of the basic system. Good working relationships are built up with users and, through working with them, the designer becomes aware of their problems. A 'spy' system logs all access requests and analysis of this usage profile prompts the development of specialized routines for common types of access. The ongoing development of the file structures leads to more efficient structures for the way the data is most commonly accessed. Users are regularly made aware of the cost of the system by printing out these costs after data retrieval. This helps to steer the user into 'better paths' and also encourages him to ask for assistance in developing new routines.

The present total size of INFO is about 2 megabytes and the typical size of a data-set is 200000 bytes. Since implementation, there have been 3 major redesigns and a host of modifications. Typically, an experienced APL-er can put up a new data-set in a couple of hours, including the basic keywords and functions. Then usage will be carefully watched for a while so that optimisation of the new system can take place. After a new data-set has settled down, its use will be checked monthly for the pattern of use to determine if further development is necessary.

This system is a shining example of 'how one ought to do it' and I hope we're going to see many more like it in the future. Congratulations to BNF APL-ers.

STAPL, the Sigplan Technical Committee on APL, (part of ACM) is a thriving organization, currently boasting about 1200 members, whose major activities are the sponsoring of APL conferences and the quarterly publication of APL Quote Quad (sent to all members of STAPL). The organization plans to sponsor an international APL conference in 1979, to support creation of local chapters of STAPL, to encourage regional and special-interest APL meetings, and to upgrade the quality of APL Quote Quad.

A new executive of STAPL has recently been elected:

Chairman:	**Phil Abrams**	STSC
Vice Chairman:	**Lib Gibson**	I.P. Sharp Associates
Secretary Treasurer:	**Martin Sandfelder**	IBM
Board of Directors:	**Mike Jenkins**	Queen's University
	Eugene McDonald	IBM
	Dick Orgass	U. of Arizona
Past Chairman:	**Garth Foster**	U. of Syracuse
APL Quote Quad Editor:	**Marilyn Pritchard**	STSC

Interested SHARP APL users should note that you may join STAPL without paying for a full ACM membership as long as your primary professional loyalty is to some discipline other than computing. Applications for membership should be made to ACM/STAPL, and mailed to:

ACM, Inc.,
P.O. Box 12105,
Church St. Station,
New York, N.Y. 10249

FILE CONVERSION

Gordon Ross
Margo Harvie

Users are often confronted with having to convert an external file to a SHARP APL file, or vice versa. File conversions are carried out by the I.P. Sharp operations staff, but it is up to the user to provide the operator with enough information about a particular file so that the conversion can be carried out successfully.

Several types of file conversions can be carried out by standard Sharp utilities. This is generally the case if the external file is a simple sequential file (in which the records consist uniformly of elements of those types represented in APL). There are four common types of file conversions that are usually handled by one of the utility programs:

1) External file (fixed blocked) to SHARP APL file
2) External file (unblocked) to SHARP APL file.
3) SHARP APL file to external file (fixed blocked).
4) SHARP APL file to external (unblocked) file.

It is important to establish whether or not the external file is (or is to be) fixed blocked. This fact will determine the choice of component and data types, and the availablility of the special translate option.

External File to SHARP APL File - Please let the operator know:

1) If the external deck is on cards, and if so, how the card deck will be identified;
2) If the external file is on tape, the
   - volume serial number
   - label
   - tape density
   - logical record size
   - block size;
3) What data types are stored on the external file;
4) What shape the APL components are to have;
5) What data type is to be used for the APL file;
6) Whether character translation is required;
7) What the account number and file name for the APL file will be;
8) The name, address and telephone number of the file owner.

SHARP APL to External Files - Please let the operator know:

1) The account number and file name for the SHARP APL file;
2) What data type should be used in the conversion (if the data is not being converted as M-entries);
3) Whether character translation is required;
4) If the file is to be output on cards, is the deck to be interpreted;
5) If the file is to be output on tape, specify
   - logical record size
   - block size
   - labels
   - density
   - who will provide the tape (user or Sharp, rented or bought);
6) The name, address and telephone number of the file owner.

FILE CONVERSION (continued)

This information should be relayed to the operator by means of the)*OPR* message, by *MAILBOX* (code *OPR*) or by contacting your SHARP APL representative. If the user is unable to supply some of the technical data specified above, a printout of the contents of the first few records of the file will help the operator to come up with the information needed.

External files other than simple sequential can also be converted to SHARP APL files. This process tends to be more complicated and may require a specifically tailored program, or post processing in APL. For further information on standard and non-standard file conversions, see the manual **File Conversions**, or contact your Sharp representative.

Note to Users of the SHARP APL Statistical and Econometric Libraries

Andrew North, Ottawa.

In February, 1975, I.P. Sharp Associates and the Institute for Policy Analysis of the University of Toronto published a book entitled **The Use and Misuse of Econometrics (with reference to SHARP APL)**. The text was designed to serve as a guide to both the neophyte and veteran econometrician, and as an illustration of the practical application of the SHARP APL statistical libraries to econometric analysis. Significant enhancements to these libraries have prompted the publication of a second edition by the original authors, David K. Foot of the Institute for Policy Analysis, University of Toronto, and Andrew North of I.P. Sharp Associates (Ottawa). Improvements include new sections outlining the application of such analyses as Shiller lags and generalized and three-stage least squares, as well as a more comprehensive discussion of the contents of the first edition. Examples are provided of sample executions of the latest versions of all pertinent SHARP APL statistical library routines.

The text is the first of a series of three econometric volumes currently available from the University of Toronto or I.P. Sharp Associates, namely:

The Use and Misuse of Econometrics (with reference to SHARP APL), 2nd Edition, by David K. Foot and Andrew North
Applied Econometrics, by Dale Orr
Exercises in Applied Econometrics, by Dale Orr and Gerry Slusar.

Dr. Orr is a Senior Policy Analyst for the Dept. of Consumer and Corporate Affairs in Ottawa, while Mr. Slusar is Acting Director, ANSSIR Division, Welfare Information Systems Branch, Health and Welfare Canada.

The contents of the Orr text are similar to those of Foot and North, the emphasis on the application of the econometric techniques to policy analysis being greater in the former. The two are intended to be complementary. The Orr-Slusar exercise manual is a companion volume to the Orr text and contains a series of practical problems, all of which are solved with the help of the econometric analysis routines in the SHARP APL public libraries.

For more information concerning any of the above texts, please contact your local SHARP APL representative.

SHARP APL COURSES

Introduction to APL:	NOV	DEC	JAN	FEB	MAR
CALGARY (5-day)			10,11,18,25→Feb. 1		
MINNEAPOLIS		06-08		07-09	14-16
MINNEAPOLIS		13-15		14-16	21-23
OTTAWA	14-18	05-09	09-13	06-10	06-10
ROCHESTER	21-25		23-27	20-24	20-24
SEATTLE				14-16	
TORONTO		12-14	03-05	13-15	06-08
TORONTO			23-25		27-29
U.K. (LONDON)	28-30		09-11	13-15	
VANCOUVER	30	→ 2	11-13	08-10	
WINNIPEG	21-24		23-26	20-23	

Intermediate

MINNEAPOLIS (6 half-days over 2 wks)	Jan. 10-12,17-19
SEATTLE	Nov. 10-11
TORONTO	On demand
WINNIPEG	On demand

Advanced

U.K.	"APL System Design"	Jan. 12-13	Mar. 9-10

Seminars

CALGARY:	"Designing SHARP APL Data Bases"	Jan. 17
EDMONTON:	"SHARP APL Public Data Bases "	Dec. 15
	"Good APL Programming Techniques"	Jan. 19
	"Regression Analysis in APL"	Feb. 17
MINNEAPOLIS:	"APL in Financial Modelling"	On demand
	"What is APL?"	On demand
SEATTLE:	"MAGIC"	Jan. 16-17
TORONTO:	"MAGIC for Time Series Analysis"	Nov. 14
	"Saving Money with NTASKS & BTASKS"	Jan. 30
	"AIDS" (3-day)	Dec. 5-7
	"Box-Jenkins"	On demand
U.K. (LONDON):	"MAGIC for Time Series Analysis"	On demand
	"Appreciaton of APL"	Nov. 17 and Feb. 2
	"Introduction to the SHARP APL system"	On demand
VANCOUVER:	"APL User Seminars"	Monthly.

APPLICATIONS SOFTWARE LIBRARY - UPDATE: Please incorporate the new or updated versions of existing packages into user applications. Contact your Sharp representative for assistance at any time.

NEW WORKSPACES

1	*NEWS*	New version - main function is *NEWS*.
3	*GRAFPLOT*	See page 1 and 2.
5	*IBM*	Library 5 is reserved for Terminal Support. 5 *IBM* contains Selectric diagnostic functions (formerly 9 *TEST*).

WORKSPACES REMOVED

3	*DEMOONLY*	Obsolete graphics demos.
3	*LPLOT*	Obsolete - alternatives are 3 *PLOT* and 3 *GRAFPLOT*. (Moved to 499 *LPLOT*3 .)
13	*DPDEMO*	Old, unused workspace, now moved to 499 *DPDEMO*3.

SIEMENS SUCCUMBS TO APL

Laurie Howard,
Intersystems, B.V.
Amsterdam.

After providing APL on limited release for a number of years, Siemens A.G., the West German computer manufacturer, has adopted APL as a major software product.

The initial implementation of Siemens APL was made in 1972/73 by a team from I.P. Sharp Associates, Toronto, and Intersystems, Amsterdam. Since that time, Intersystems, the Dutch subsidiary of I.P. Sharp Associates, has been retained by Siemens to enhance the product and implement the latest language features.

Siemens APL runs as a non-dedicated system, under the BS 2000 operating system, on the Siemens 4004 and 7000 series machines. The system contains most major features of SHARP APL, including the file subsystem. Current development work in Amsterdam will result in a new release, scheduled for mid 1978, with a very high degree of compatibility with SHARP APL.

Sharp Special Systems

UPDATE ON MAPL

Bob Johnston
Audrey Sharp

MAPL, the minicomputer-based multi-terminal front end, designed for use with the SHARP APL system, was first discussed in the newsletter in June.

Development of the MAPL concept has continued through the summer months, and a demonstration of the (then current) system was given to the I.P. Sharp Branch Managers when they convened in Toronto in October. As a result of the questions which arose at that session, we realized the time had come to issue more detailed and specific information about MAPL. A "MAPL Questions and Answers" brochure has been produced, which should be available from your local Sharp representative at about the same time as this newsletter.

The brochure deals with many aspects of MAPL, from the purely technical (How far away from my PDP-11 can I put my terminals?) to the philosophical (What are the advantages of using MAPL rather than an in-house mini?) We have also attempted to report on the 'State of the Art', though this changes weekly. Basically, MAPL can now handle multi-terminal operation, with line-at-a-time output of multiple-line output from APL, automatic sign-on and time-out.

By the time this newsletter is released, the option for identifying terminals (and thus having the APL application program direct output to selected terminals) should be available - together with a suitable APL package to interact with MAPL in this way.

The next phase will be to support formatted CRT's - and we are interested in any suggestions you may have on that subject.

I.P. Sharp Associates **Head Office:** 145 King Street West, Toronto, Canada M5H 1J8 (416) 364-5361

INTERNATIONAL BRANCH OFFICES

Amsterdam
Intersystems B.V.
Herengracht 244
1016 BT Amsterdam
The Netherlands
(020) 24 40 50

Birmingham
I.P. Sharp Associates Limited
2nd Floor, Radio House
79/81 Aston Rd. North
Birmingham B6 4BX
England
021-359-6964

Boston
I.P. Sharp Associates, Inc.
Suite 812
148 State St.
Boston, Mass. 02109
(617) 523-2506

Brussels
I.P. Sharp Europe S.A.
Ave. General de Gaulle, 39
1050 Bruxelles, Belgium
(02) 649 99 77

Calgary
I.P. Sharp Associates Limited
Suite 2660, Scotia Centre
700 - 2nd St. S.W.
Calgary, Alberta T2P 2W2
(403) 265-7730

Chicago
I.P. Sharp Associates, Inc.
2 North Riverside Plaza
Room 1746
Chicago, Illinois 60606
(312) 648-1730

Copenhagen
I.P. Sharp ApS
Østergade 24B
1100 Copenhagen K
Denmark
(01) 112 434

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

Düsseldorf
I.P. Sharp GmbH
Leostrasse 62A
4000 Düsseldorf 11
West Germany
(0211) 57 50 16

Edmonton
I.P. Sharp Associates Limited
Suite 505, 10065 Jasper Ave.
Edmonton, Alberta T5J 3B1
(403) 428-6744

Gloucester
I.P. Sharp Associates Limited
29 Northgate St.
Gloucester, England
0452 28106

Houston
I.P. Sharp Associates, Inc.
Suite 405, One Corporate Square
2600 Southwest Freeway
Houston, Texas 77098
(713) 526-5275

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

London, England
I.P. Sharp Associates Limited
132 Buckingham Palace Rd.
London SW1W 9SA
England
(01) 730-0361

Minneapolis
I.P. Sharp Associates, Inc.
Suite 1371, 1 Appletree Square
Bloomington, Minn. 55420
(612) 854-3405

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

New York City
I.P. Sharp Associates, Inc.
Suite 250, East Mezz.
Pan Am Bldg.
New York, N.Y. 10017
(212) 986-3366

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

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 110, 299 California Ave.
Palo Alto, Ca. 94306
(415) 327-1700

Rochester
I.P. Sharp Associates, Inc.
Suite 1150, 183 Main St. E.
Rochester, N.Y. 14604
(716) 546-7270

San Francisco
I.P. Sharp Associates, Inc.
Suite C409, 900 North Point St.
San Francisco, Ca. 94109
(415) 673-4930

Seattle
I.P. Sharp Associates, Inc.
Suite 217, Executive Plaza East
12835 Bellevue-Redmond Rd.
Bellevue, Wa. 98005
(206) 453-1661

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

Toronto
I.P. Sharp Associates Limited
145 King Street West
Toronto, Ontario M5H 1J8
(416) 364-5361

Vancouver
I.P. Sharp Associates Limited
Suite 604, 1112 West Pender St.
Vancouver, B.C. V6E 2S1
(604) 682-7158

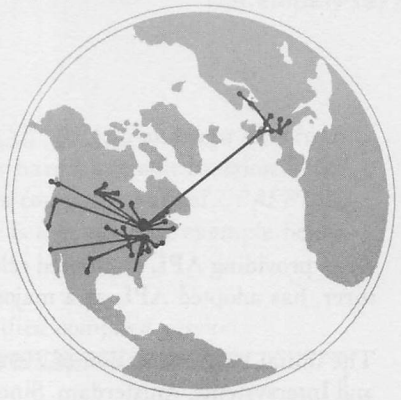
Vienna
I.P. Sharp Ges. MbH
Rechte Wienzeile 5/II
3 Wien, Austria

Warrington
I.P. Sharp Associates Limited
48A Horsemarket Street
Warrington, Cheshire
England
(020) 55342

Washington
I.P. Sharp Associates, Inc.
Suite 307, 1730 K Street N.W.
Washington, D.C. 20006
(202) 293-2915

Winnipeg
I.P. Sharp Associates Limited
Suite 909, 213 Notre Dame Ave.
Winnipeg, Manitoba R3B 1N3
(204) 947-1241

Zürich
I.P. Sharp A.G.
Badenerstrasse 141
8004 Zürich
Switzerland
241 52 42



APL Operator Voice (416) 363-2051
Communications (416) 363-1832

SHARP APL Local Access In:

Canada
Calgary
Edmonton
Halifax
Hamilton
Kitchener
London
Montreal
Ottawa
Quebec City
Regina
Saskatoon
Toronto
Vancouver
Victoria
Winnipeg

U.S.A.
Ann Arbor
Boston
Buffalo
Chicago
Dallas
Dayton
Des Moines
Houston
Los Angeles
Minneapolis
Newport Beach
New York City
Raleigh, N.C.
Rochester, N.Y.
San Francisco
Seattle
Stamford, Ct.
Syracuse, N.Y.
Washington, D.C.
White Plains

Europe
Amsterdam
Birmingham
Bruxelles
Copenhagen
Coventry
Düsseldorf
Gloucester
Liverpool
London
Manchester
Milan
Paris
Stockholm
Vienna
Zürich

TELENET AND TYMNET

Alabama
Birmingham

Arizona
Phoenix

Arkansas
Little Rock

California
Alhambra
El Segundo
L.A. Central & East
L.A. West & Valley
L.A. South & Long Beach
Mountain View
Oakland
Orange County
Oxnard
Riverside
Sacramento
San Clemente
San Carlos
San Diego
S.F. Peninsula
San Jose
Santa Ana
Santa Barbara
Santa Rosa

Colorado
Denver
Colorado Springs

Connecticut
Bridgeport
Danbury
Darien
Hartford
New Haven/Bridgeport
Waterbury

Delaware
Wilmington

Florida
Fort Lauderdale
Jacksonville
Miami
Orlando
Pensacola
St. Petersburg
Tampa

Georgia
Atlanta

Idaho
Boise

Illinois
Freeport
Rockford

Indiana
Fort Wayne
Indianapolis
South Bend

Iowa
Cedar Rapids
Iowa City

Kansas
Topeka
Wichita

Kentucky
Louisville

Louisiana
Baton Rouge
Lafayette
New Orleans

Maryland
Baltimore

Massachusetts
Cambridge

Michigan
Detroit
Jackson
Kalamazoo
Southfield

Missouri
Kansas City
St. Louis

Nevada
Carson City
Las Vegas

New Hampshire
Nashua

New Jersey
Englewood Cliffs
Moorestown
Princeton
Piscataway
Newark
Union
Wayne

New York
Albany
Corning
Hempstead
Long Island
Niagara Falls

North Carolina
Durham
Raleigh/Durham

Ohio
Akron
Cincinnati
Cleveland
Columbus
Toledo

Oklahoma
Oklahoma City
Tulsa

Oregon
Portland

Pennsylvania
Allentown
Erie
Harrisburgh
Philadelphia
Pittsburgh
Valley Forge
York

Rhode Island
Providence

South Carolina
Greenville

Tennessee
Memphis

Texas
Austin
Baytown
Beaumont
El Paso
Ft. Worth
Midland
Odessa
San Antonio

Utah
Salt Lake City

Virginia
Norfolk
Richmond

Wisconsin
Madison
Milwaukee
Oshkosh

☐ Please amend my mailing address as indicated.
☐ Add to your mailing list the following name(s).
☐ Send me a SHARP APL publications order form.

Name: _____

Co.: _____

Address: _____

The Newsletter is a regular publication of I.P. Sharp Associates. Contributions and comments are welcome and should be addressed to: Jeanne Gershater, I.P. Sharp Newsletter, 145 King Street West, Toronto, Canada M5H 1J8.