the I.P.Sharp newsletter

AUGUST/SEPTEMBER 1975

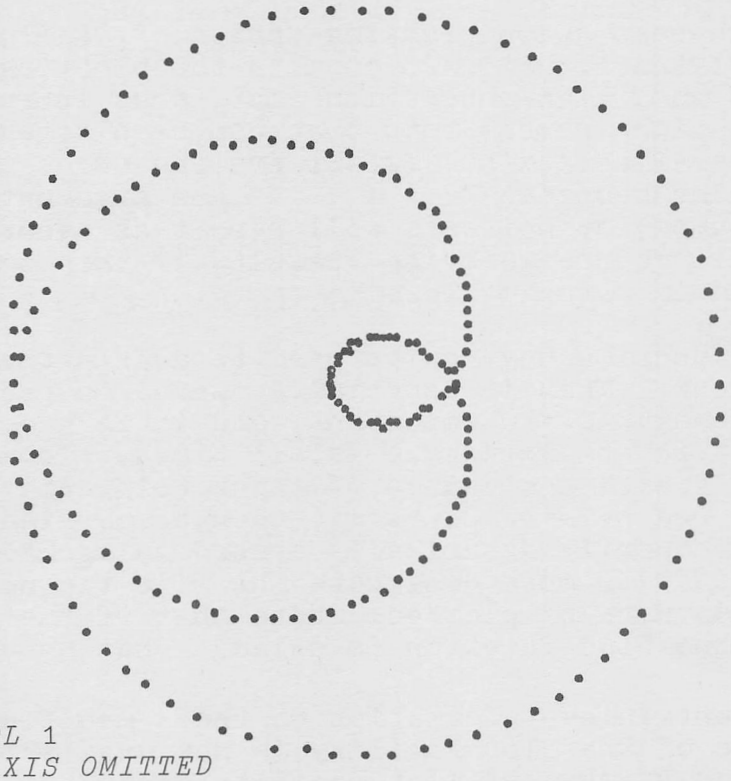
The Plot thickens...

CHANGES TO THE *SHARP APL*

Plot Facility

by Leslie Goldsmith

The graph below demonstrates the sophisticated plotting facility provided by *SHARP APL*. The results of data manipulation can be plotted quickly for a variety of applications.



```
FINEPLOT
ABSCISSACOL 1
ORDINATE AXIS OMITTED
ABSCISSA AXIS OMITTED
THETA←(0,ι400)×○÷20
PLOT ⍉(2 401 ρ4×2○THETA×0.1)× 2 1 ∘.○THETA×0.5
```

Graphs are generated using the standard *APL* typeball, or the *FINEPLOT* typeball which, with its special set of symbols, gives a resolution of 900 points per square inch.

CHANGES TO THE *SHARP APL* PLOT FACILITY

Several changes and enhancements have been made to the *SHARP APL* Plot Facility. Users who have saved previous versions of the plot package may take advantage of these modifications by copying the group *CHANGED* from 3 *PLOT* into their application workspace.

Enhancements to the Plot Facility include:

* Processing time for graphs has been reduced considerably; an average plot now takes about 17 percent less cpu time than previously.

* The core overhead used by the *PLOT* function has been decreased. This means that there is less chance of a *WS FULL* occurring while *PLOT* is executing in a workspace with many user-defined "cover functions". The storage space for *PLOT* has also been reduced significantly.

* The option *MULTIPLE*, versus its default *SINGLE*, may now be specified with *HISTOGRAM* as well as *NORMAL*. With *MULTIPLE*, when points coincide, the symbols will be overstruck. If *HISTOGRAM* is also in effect, points will be overstruck from the concurrent point downward in the output column.

* A second high-resolution plotting option, *FINEAPL*, has been added to complement *FINEPLOT*. *FINEAPL* permits the plotting of certain *APL*-like symbols on a high-resolution plot, thus increasing the total number of distinct symbol-sets that can be plotted to six. The symbol-sets available with *FINEAPL* are the three used by *FINEPLOT*, followed by the sequence o * ⊛ . If the argument to *PLOT* has more than six columns, symbol-sets will repeat as necessary. *FINEAPL* and *FINEPLOT* will produce identical results if the number of columns to be plotted is three or less.

* The *SRCFILE* function may now be used to define the *PLOT* function when convenient. This is especially useful in packages in which space considerations are important, and it is undesirable to have *PLOT* resident in the workspace at all times. To make the *PLOT* function available with the source of input being its argument, execute *SRCFILE* 0 . One or more plots may then be carried out, and the function subsequently dynamically erased using □*FD*. A non-zero argument to *SRCFILE* must designate the file tie number and component holding the data to be plotted. Note that if the source of input is from files, the *PLOT* function is niladic (has no arguments).

A new manual containing information on these modifications and the general use of the Plot Facility is now available. The manual, entitled "*SHARP APL* Plot Facility Instruction Manual" (Copyright 1975), also contains two new appendices which provide the user with easy-to-use reference information.

The first of these, "Summary of State-setting Functions", contains information on each state option, its alternatives, and a brief description of its effect. The second, "Summary of Error Reports", describes the possible errors a user might encounter while using the Plot Facility. Along with each report is listed the names of the functions in which it may occur, its cause, and a suggested remedy to the problem.

For more information on the Plot Facility or a copy of the manual, please contact your local *SHARP APL* representative.

ANOVA

ANALYSIS OF VARIANCE - LATIN SQUARE DESIGN

by ROGER HUI

A Latin Square is an MxM square array of M symbols such that each symbol occurs exactly once in each row and each column. The generation of all Latin squares of the same size that are distinct, that is, squares that cannot be obtained from each other by permuting the rows and columns, is a topic of group theory. In the area of analysis of variance, a Latin square design may be used if there are three treatments with the same number of treatment levels in each. The rows of the square would represent one treatment, the columns another, and the symbols within the square the third. Such a design offers economy of effort to experimentalists willing to accept its limitations.

To assist users of *SHARP APL* in analysing data from a Latin square design experiment, the functions *LATINSQC* and *LATINSQP* have been added to the *APLSTAT* library. The analysis performed by the two functions are identical, with *LATINSQC* accepting its inputs conversationally and *LATINSQP* accepting its in the form of arguments.

As an example (taken from Dixon and Massey, Introduction to Statistical Analysis, McGraw-Hill, 1957, pp. 172-174), suppose four lists of words from a previous dictation test are administered to four groups of children. Furthermore, the lists are presented using four tests of different types. A Latin square design is used, with the rows representing lists of words, the columns groups of children, and the symbols representing test types, as shown below:

A - multiple choice
B - second dictation
C - wrongly spelled word
D - skeleton word

Thus, the design might appear as follows:

```
     DESIGN
ABCD
DABC
CDAB
BCDA
```

so that the second group of children is administered the second group of words via a multiple choice test.

The data recorded are the numbers of words in each test that each group spelled correctly which were spelled incorrectly before.

```
     DATA
81  41  44  53
38  97  42  49
31  43  67  36
57  33  43  81
```

The function *LATINSQP* can be used to analyze this data, using this design, by typing:

```
     DESIGN LATINSQP DATA
ALIGN PAPER
```

ANALYSIS OF VARIANCE -- LATIN SQUARE DESIGN

SOURCE	*SS*	*DF*	*MS*	*F-RATIO*	*PLEVEL*
ROWS	359.5000	3	119.8333	1.1855	0.3914
COLUMNS	74.5000	3	24.8333	0.2457	0.8617
TRTMNTS	4626.5000	3	1542.1667	15.2564	0.0031
ERROR	606.5000	6	101.0833		
TOTAL	5667.0000	15			

Since only the treatments (tests) mean square is significantly large relative to the residual mean square, it can be concluded that the groups of children are of similar ability and the lists are of equal difficulty, but that the four tests which make up each list are significantly different.

LATINSQC and *LATINSQP* can be found in the public library workspace 34 *ANOVA*.

"*SHARP APL* in Financial Analysis" - The manual, which includes complete documentation of the entire financial analysis libraries, as well as examples and suggestions for their most effective use, is now available from your local *SHARP APL* representative.

SHARP APL

Technical Supplement

by Clement Kent

Although *APL* is a relatively easy programming language to learn, there are many non-obvious tricks and techniques that can help you use it efficiently. The Technical Supplement to the newsletter will present advanced techniques in programming and database management. To avoid inflicting this material on those not interested, the supplement will be separate from the newsletter, and will be available through your local Sharp representative. For this issue, however, we've included some of the simpler material in the newsletter to give some idea of what it's like.

TIMING

One problem frequently encountered in designing a system is to estimate how expensive it will be, or to decide which of several programming methods is likely to be the cheapest. Unfortunately, due to the continual upgrading of our hardware and software, it's not possible to give hard and fast rules about what's the most efficient way to do things. That's why timing functions are so important. So here we introduce some timing routines with the useful property that they can be used to time the execution of an arbitrary *APL* expression. They all use ⎕*FD*, the function definition quadfunction.

First, a function which, in one form or another, is familiar to everyone:

```
     ∇ XEQ CODE;FOO
[1]    CODE←'∇ FOO',CR,'[1]  ',CODE,'∇'
[2]    →(0=1↑0⍴CODE←3 ⎕FD CODE)⍴ERR
[3]    FOO ※ →0
[4]   ERR:'ERROR IN CODE - 3 ⎕FD RESULT: ';CODE
     ∇
```

The right argument can be almost any valid *APL* expression. It becomes the first line of the local function *FOO*, which is defined, executed and erased by *XEQ*.

```
     XEQ '2+2'
4
```

Using ⎕*FD* again, we construct a timing function for an arbitrary expression, or "code string".

```
       ∇ Y←N TIME CODE;FOO;X;I
[1]   →(×N)↓Y←0
[2]   →(0≠1↑0⍴X←3 ⎕FD '∇ FOO;FOO;X;I;N;Y',CR,'[1] ',CODE,'∇')⍴START
[3]   'ERROR IN CODE - 3 ⎕FD RESULT: ';X ※ →0
[4]  START:I←1 ※ X←⎕AI[2]
[5]  LOOP:FOO ※ →(N≥I←I+1)⍴LOOP
[6]   →(×Y)⍴END ※ Y←⎕AI[2]-X
[7]    6 ⎕FD 'FOO' ※ X←3 ⎕FD '∇ FOO',CR,'[1] X←X∇' ※ →START
[8]  END:Y←⌊0.5+(Y-⎕AI[2]-X)÷N
     ∇
```

N is the repetition factor - the number of times the expression is executed. The result *Y* is the average time per execution, in cpu units.

The question of how accurate this timing is, is a difficult one. One way of testing this function is to run it for the same code string with ever larger numbers of repetitions, and see if the results converge to some value.

```
    ∇ Y←N STABILITYTEST CODE
[1]   Y←⍳0
[2] LP:→(×⍴N)↓0
[3]   Y←Y,N[1] TIME CODE
[4]   N←1↓N ⋄ →LP
    ∇
      (REP←10×⍳10) STABILITYTEST 'Z←⎕READ 1 1'
34  0  ¯4  36  14  ¯5  ¯2  19  15  29

      REP STABILITYTEST 'Z←⍉20 20⍴''⍺'' '
46  41  44  43  43  46  44  46  45  46
```

Quite clearly, the test is not stable when applied to the file reading expression, but it does seem to provide a reasonable answer for the monadic transpose. This points out an important point: the variation in the execution time of an expression is as important as the mean time of execution, and must be evaluated. Note that, even if we go up to 1000 repetitions in timing the file read, the time we get is not good. This is mostly due to the fact that the time required to execute the file read is comparable to the time required to execute the loop containing the *FOO* function, so that the variability in time of execution of the loop degrades our estimate of the time for the read. (In addition, there are some peculiarities in timing specific to file operations; it is better to code a routine specifically to time these operations than to use a general one.) As a rule, the best one can say is that, if the difference between the timings of a large number of repetitions of two expressions is greater than 10 or 20 percent, then there is a significant difference.

An interesting use of the above timing functions is testing the dependence of the time to execute an expression on the size of its arguments. In the example below, we timed the transpose of a character matrix for matrices of sizes 10 x 10 to 140 x 140. The program *SIZETEST* is a bit more complicated than it need be for this problem; but it can be used to time arbitrary expressions with arguments of arbitrary type and rank, and so is perhaps a useful utility.

```
    ∇ Y←N SIZETEST RHO;R
[1]   ⍝TYPE, VARNAME ARE GLOBALS
[2]   Y←⍳0
[3] LP:→(×R←1↑⍴RHO)↓0
[4]   VARNAME ASSIGN (,(R↑1)/[1] RHO)⍴TYPE
[5]   Y←Y,N TIME CODE    ⍝CODE IS GLOBAL
[6]   RHO←((⍴⍴RHO)↑1)↓RHO ⋄ →LP
    ∇

    ∇ NAME ASSIGN VALUE
[1]   XEQ NAME,'←VALUE' ∇

   CODE←'A←⍉B' ⋄ VARNAME←'B' ⋄ TYPE←'⍺' ⋄ RHO←⍉2 14⍴10×⍳14
   ⎕←Y←20 SIZETEST RHO
11  40  89  158  246  353  482  628  795  982  1191  1430  1677  1944
```

The space occupied by the matrices in this test was *X←×/RHO*.

The reader may wish to verify the assertion that the most reasonable polynomial approximation to *Y* as a function of *X* is the quadratic whose coefficients are given by:

```
      □←C2←Y⌹X∘.*0 1 2
1.288 0.097 9.643E¯8
```

which implies that the cost of a transpose is linearly related to the size of the matrix, when the space occupied by the matrix is less than 15,000 bytes. Please note that this assumes we are in a fairly empty workspace.

Does the cost of a transpose depend on the type of the matrix?

360/75 *TIMINGS FOR* 40×40 *MATRICES*

OPRN↓TYPE→	*CHAR*	*BOOL*	*INTG*	*REAL*
ϕ	134	169	134	159
⊖	140	168	128	151
⍉	167	204	168	180

It is clear that, for some reason, the transposing of an integer matrix is easy, while transposing a boolean matrix is harder. This is especially interesting in the light of the linear dependence of time on space in bytes that we found in character matrices. Note that the character matrix takes up 1600 bytes, the boolean 200, the integer 6400, and the real 12,800, which makes the real transpose about 70 times faster per byte than the boolean. This is due to the way information is physically stored and retrieved in IBM machines, in 4-byte "words". Getting a word is easy, but getting into a word is harder. Booleans are compact, since you can get 32 of them into a word, but if you have to reshuffle them it can take a while. The same thing applies to characters, to a lesser extent.

360/75 *TIMINGS FOR* 40×40 *MATRICES*

OPRN↓TYPE→	*BOOL*	*INTG*	*REAL*
+/[1]	211	19	19
+/[2]	10	18	19
+\[1]	71	27	45
+\[2]	69	29	39

2×1000 *MATRICES*

OPRN↓TYPE→	*BOOL*	*INTG*	*REAL*
	261	24	27
	4	23	25
	95	36	53
	97	36	35

Here we notice that +/ is slow in booleans, except when it goes along the last dimension, where it is very fast. This surprising result is the basis of a very efficient cross-tabulation package due to be released in the next few months.

Suppose we have a number of questions with multiple choice answers. The questions were answered by a group of respondents, and we wish to cross-tabulate their responses in a number of ways. If the results for each question are stored in a boolean matrix with as many rows as choices, and as many columns as respondents, then the following program provides a very simple way of cross-tabulating the results:

```
    ∇ Y←A AND B;R1;R2;I;L
[1]   →(((I←¯1↑R1)=¯1↑R2)∧∧/2≤(⍴R1←⍴A),⍴R2←⍴B)⍴L1
[2]   CR,'INVALID ARGUMENT SIZES FOR <AND>',CR ⋄ →
[3] L1:Y←(0,R2)⍴0 ⋄ A←((L←×/R1←¯1↓R1),I)⍴A ⋄ I←1
[4] L2:Y←Y,[1] B∧R2⍴A[I;]
[5]   →(L≥I←I+1)⍴L2 ⋄ Y←(R1,R2)⍴Y
   ∇
```

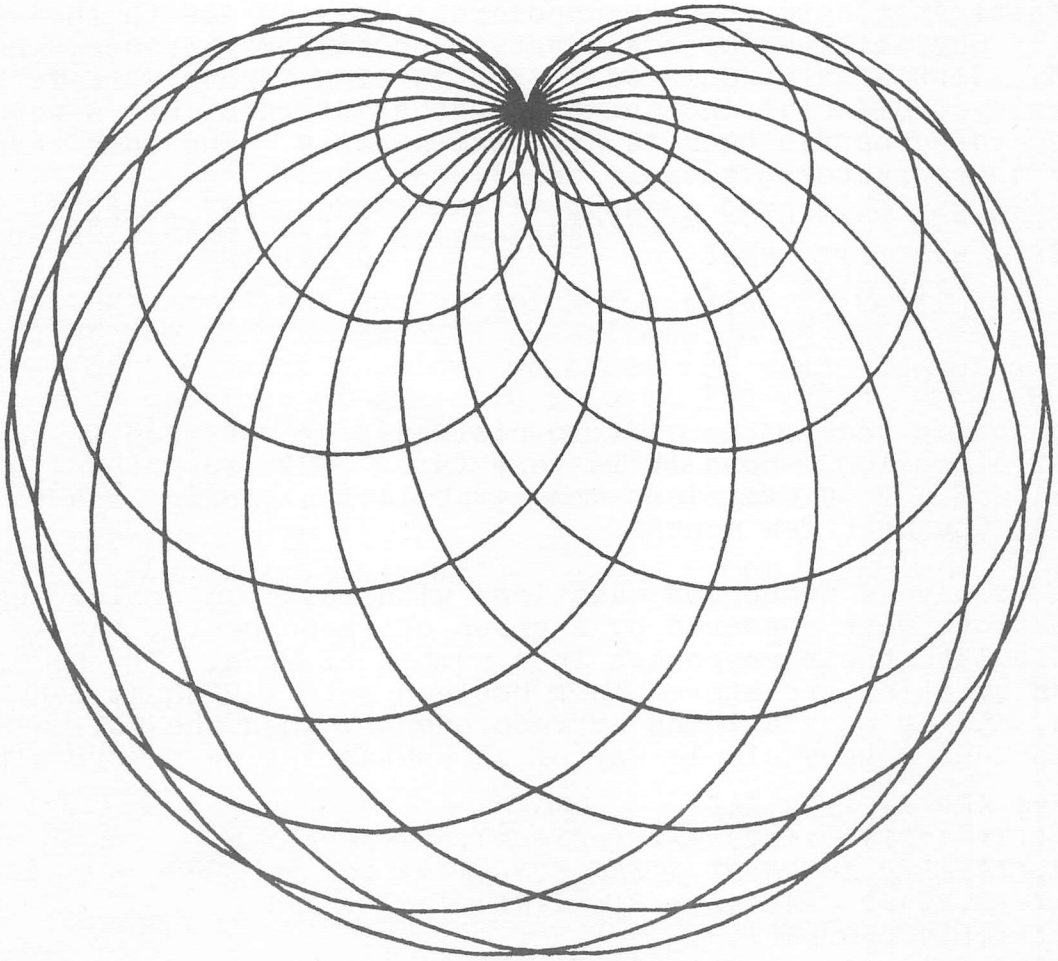
If each question has a 1 in a row when the respondent in that column answered with that choice, then the following expression gives the cross-tabulation:

```
+/ QUESTION1 AND QUESTION2 AND QUESTION3 ... AND QUESTION X
```

These raw scores can be used in reporting or statistical routines to provide sophisticated market survey analysis packages.

Next issue we will discuss functions on files, restartable code, and interesting applications of scan. Questions, comments and contributions are welcomed and should be addressed to:

Clement Kent, (mailbox code *KENT*)
I. P. Sharp Associates Limited,
Suite 1400,
145 King Street West,
Toronto, Ontario M5H 1J8.



EQUIPMENT CHANGES

In the last issue of the newsletter we discussed the equipment changes being made in the computer room and stated that most of these would go unnoticed by our customers.

The changes are proceeding more or less on schedule, maybe a week or two behind our original schedules. The first 360 model 75 was installed during August and around the middle of September it went on-line, and the first 370 model 145 was returned to IBM. The two systems were merged onto a single model 75 in late September and the second 145 was returned to IBM. The second model 75 is due to be delivered in early October and should be operational by the middle of the month.

3705 programming was completed by Roger Moore in an astonishingly short time and it has been on-line since mid-September. The 3705 now provides a contention system under which all the local lines from all cities on the network contend for a limited number of access "ports" on the model 75. Work is proceeding on provision of support for high speed synchronous input to the 3705 and as soon as this is complete the first packet switching facility (from London, England) will be installed. Subsequently all the high density communications routes will be changed to packet switching.

There is no progress as yet on the double-density direct access storage devices; hopefully we will have more news to report in the next issue.

We have experienced a few teething troubles in settling in the model 75, but most of these now appear to be behind us. We are using two high-speed drums as swapping devices and commissioning them took longer than expected.

NEW NEWSLETTER

The continued success of David Keith's *MAGIC* system applied particularly to the CAB airline data-base, has prompted us to start a new newsletter especially for users of that system.

MAGIC is a language designed specifically to retrieve and manipulate time series data. Shortly we will be making available the OECD data including principal economic indicators for all OECD countries. Users of *MAGIC* and anyone else interested in CAB, OECD, NBER or other data in time series form can arrange to receive the Aeronautics Newsletter by requesting it from their closest Sharp office.

IF IT'S TUESDAY IT MUST BE BOSTON (OR MONTREAL)!

We have been giving a series of intensive two-day *APL* seminars in 14 of our offices. The mix of slides, notes, discussions, case studies and sessions at the terminal seems to be a highly successful way of introducing neophytes to *APL*. In spite of the hectic travelling schedule and the problem of having more suitcases than hands, the instructors seem to enjoy the smiles saying "well, now it makes sense!"

The seminars include discussions of most *APL* primitives, some operators, a smattering of the file subsystem and an important section on the valid and invalid contexts for using *APL*. Case studies are an effective means of forcing the student to organize his limited knowledge of *APL* to tackle an extended problem. This seems superior to the staccato responses to problem sections found in most *APL* texts.

If you have missed the seminar in your area and are prepared to spend the two days (yes, there is homework), please contact your local *SHARP APL* representative. After the last of the scheduled courses in Dallas and San Francisco on October 1st. and 2nd., course materials will be available for additional two-day seminars.

Alan Daley

SHARP APL COURSES

Three- or five-day Introduction to *APL* courses are offered at our offices in:

LONDON, ENGLAND		November 12-14, 17-18
LOS ANGELES	October 6-10	November 3-7
MONTREAL		November 3-7
OTTAWA	October 6-10	November 3-7
ROCHESTER	October 20-24	November 17-21
TORONTO week of	October 14th and	November 10th
VANCOUVER	October 6-8	

A series of one-day seminars are also being held in London, England, as follows:

SHARP APL File Management	- October 2nd
APL and its Commercial Uses	- October 16th
SHARP APL for Financial Planners	- November 6th.

Please contact your local *SHARP APL* representative if you are interested in attending a course.

SHARP APL MANUALS and PRODUCT LITERATURE - ORDER FORM

Any one or more of the following publications may be ordered prepaid by completing the form below and mailing it to your local Sharp office, or:

I. P. Sharp Associates Limited,
Suite 1400,
145 King Street West,
Toronto, Ontario M5H 1J8.

Date: _____

Code	Title	Qty.	Price	Total
006SA01	A Breakthrough in the Computer Art (brochure)		N/C	
007SA01	The Problem Solvers (brochure)		N/C	
090SA06	*SHARP APL* Reference Card		N/C	
095SA01	Orange Three Ring Binders (for manuals)		5.00	
110SA04	An Introduction to *SHARP APL*		2.00	
115SA01	A *SHARP APL* Minicourse		2.00	
125SA02	An Actuarial Introduction to *SHARP APL*		2.00	
150SA03	*SHARP APL* Library Documentation		10.00	
210SA05	*SHARP APL* File Subsystem Instruction Manual		2.00	
215SA04	*SHARP APL* Plot Facility		2.00	
220SA05	*SHARP APL* Report Formatting		2.00	
225SA04	Linear Programming in *SHARP APL*		2.00	
310SA03	*SHARP APL* Massager Plus		4.00	
350SA02	*SHARP APL* Financial Post		4.00	
710SA02	*SHARP APL* in Financial Analysis		2.00	
750SA01	*SHARP APL* in Statistical Analysis (brochure)		N/C	
760SA01	*SHARP APL* Functions for Statistical Analysis		10.00	
410SA02	An Introduction to *AIDS*		2.00	
425SA02	*AIDS* Reference Manual		15.00	
810SA02	*SHARP APL* Skills Inventory System		2.00	
820SA01	The Use and Misuse of Econometrics (With Reference to *SHARP APL*) by - David K. Foot and Andrew North		5.00	

FROM: _____ SHIP TO: _____

_____ _____

_____ _____

_____ _____

Signed: _____

Suite 1400, 145 King Street West, Toronto, Canada M5H 1J8

Update

☐ Please amend my mailing address as indicated.

☐ Add to your mailing list the following name(s).

☐ Send me SHARP APL manuals and product literature as listed.

☐ Note my comments: _____

The Newsletter is a regular publication of I.P. Sharp Associates Limited. Contributions and comments are welcomed and should be addressed to: Jeanne Gershater, Editor, I.P. Sharp Newsletter, Suite 1400, York Centre, 145 King Street West, Toronto, Ontario, M5H 1J8.

I.P. Sharp Associates Limited Head Office: Suite 1400, 145 King St. West, Toronto, Canada M5H 1J8 (416) 364-5361

Canada — Regional Offices

Calgary
Suite 1000,
615 — 2nd Street S.E.,
Calgary, Alberta
T2G 4T8
(403) 265-7730

Ottawa
Suite 2003,
210 Gladstone Avenue,
Ottawa, Ontario
K2P 0Y6
(613) 236-9942

Vancouver
Suite 604,
1112 West Pender St.,
Vancouver, B.C.
V6E 2S1
(604) 682-7158

Montreal
Suite 1610,
555 Dorchester Blvd. West,
Montreal, Quebec
H2Z 1B1
(514) 866-4981

London
Suite 1400,
275 Dundas St.,
City Centre, Canada Trust Tower,
London, Ontario
(519) 434-2426

England

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

London
118 - 119 Piccadilly,
Mayfair, London W1V 9FJ
England
(01) 629-1564

Europe

Intersystems, B.V.
Herengracht 244,
Amsterdam 1002,
The Netherlands
(020) 250401

APL Europe S.A.
Ave. du Général de Gaulle 39,
105 Bruxelles,
Belgium
(649) 94 30

U.S.A. — I.P. Sharp Associates, Inc.

Boston
21 Merchants Row,
Boston,
Mass. 02109
(617) 523-2506

Newport Beach
Suite 1135,
610 Newport Centre Drive,
Newport Beach, Ca. 92660
(714) 644-5112

Rochester
Suite 1150,
183 Main Street East,
Rochester, N.Y. 14606
(716) 546-7270

San Francisco
Suite C409,
900 North Point Street,
San Francisco, Ca. 94109
(415) 673-4930

New York
Suite 39, Station Plaza,
250 East Hartsdale Ave.,
Hartsdale, N.Y. 10530
(914) 472-6380

Chicago
Suite 424,
8501 West Higgins Rd.,
Chicago, Ill. 60631
(312) 693-5895

Minneapolis
Suite 104,
5001 Cedar Lake Road,
St. Louis Park,
Minn. 55416
(612) 374-9406

Dallas
Suite 1148,
Campbell Centre,
8350 Northcentral Expressway,
Dallas, Texas 75206
(214) 369-1311

Seattle
10105 Marine View Drive
Seattle, Wa. 98146
(206) 935-2867

Washington D.C.
1815 Fort Myer Drive
Arlington, Va. 22209
(716) 546-7270 - Rochester

PRODUCTS DIVISION:

Canada
Box 1900,
150 Rosamond Street,
Carleton Place, Ontario
K0A 1J0
(613) 257-3610

U.S.A.
Bridge Administration Building,
Bridge Plaza,
Ogdensburg, New York 13669
(315) 393-0733

SHARP APL Local Access In:

Canada
Calgary
Edmonton
Halifax
Kitchener
London
Montreal
Ottawa
Quebec City
Regina
Sault Ste. Marie
Toronto
Vancouver
Winnipeg

U.S.A.
Atlanta
Boston
Buffalo
Chicago
Cleveland
Dallas
Los Angeles
New York City
Rochester
San Francisco
Santa Ana
Seattle
St. Paul, Minn.
Syracuse
Washington
White Plains

Europe
London
Gloucester
Amsterdam
Düsseldorf