JANUARY / FEBRUARY 1983
Volume 11 Number 1

I.P. Sharp newsletter

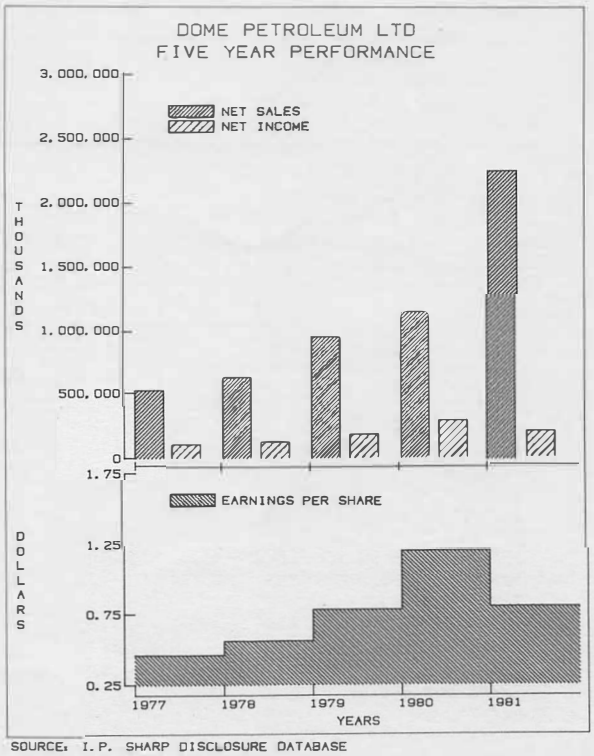
DISCLOSURE II
Data Base

I.P. Sharp now offers you an excellent means of keeping abreast of competition, investment opportunities, and marketing and financial conditions, with the DISCLOSURE II data base. You can also track trends in executive compensation, market distribution, and profitability of any group of public companies.

The DISCLOSURE II data base includes current and historical statistics and textual information for 8 500 publicly traded corporations that have filed reports with the U.S. Securities and Exchange Commission (SEC). Most of these companies' stocks are traded on the New York or American Stock Exchange, or over the counter.

DISCLOSURE II is of particular interest to financial advisors, money managers, corporate planners, investment analysts, stockbrokers and others involved in money management, accountants, attorneys, marketing analysts, and librarians.

You can get data for each company in numeric or textual form. The numeric portion contains annual data from balance sheets (assets and liabilities for the past two years) and from income statements (for the past three years). Quarterly data includes income statements for the current filing year. A five-year summary is available for sales, net income, and earnings per share. In addition, I.P. Sharp calculates and stores many key financial ratios in the data base.



SOURCE: I. P. SHARP DISCLOSURE DATABASE

in this issue

. . . DISCLOSURE II Data Base

The textual portion includes company name and address, names, ages, titles, and annual remuneration for officers and directors, filing tables, auditor's reports, ownership, subsidiaries, special exhibits, corporate events, and the full text of the management discussion as it appears in the company annual report. As well, segment data (sales and operating income for major lines of business) is available.

The data base is updated weekly with relevant changes, using data supplied by Disclosure Incorporated of Bethesda, Maryland.

Retrieving data

You can retrieve data from DISCLOSURE II by specifying DISCLOSURE number, CUSIP number, or company number for the company or companies desired. In addition, you can screen the data base to look for companies which satisfy specific search criteria. For example, you can retrieve simply and efficiently all airlines in the Standard Industrial Classification (SIC) code 451, with assets of more than a billion dollars. *(See figure 1)*

You can combine data from DISCLOSURE II with data from other I.P. Sharp data bases, such as U.S. Stock Options, U.S. Stock Market, and Money Market Rates.

You can retrieve, manipulate, and format easily the data exactly the way you want to see it, using MAGIC, I.P. Sharp's data retrieval and analysis software package. For example, you can use MAGIC to retrieve and print all income statement items for a company for the past two years, and also get the year-over-year percentage change.

You can display data from the DISCLOSURE II data base as a multicolour plot, chart, or graph with SUPERPLOT, I.P. Sharp's business graphics software. The SUPERPLOT on the front page of the *Newsletter* displays the five-year summary of earnings per share, net sales, and net income from the annual data of the DISCLOSURE II data base for Dome Petroleum.

There is no upfront subscription fee for the use of the DISCLOSURE II data base. You are charged a royalty fee only for the items of information actually retrieved. Frequent users may decide to subscribe to the data base for a fixed annual fee without incurring further royalty charges■

Figure 1

```
      RESETOPTIONS
      NOTIMESERIES
      AUTOLABEL
      YEARLY DATED AT 81
      TITLE 'U.S. AIRLINES WITH ASSETS OVER $1 BILLION (1981)'
      FOOTNOTE 'SOURCE: DISCLOSURE II DATA BASE'
      SCREEN←'/Q/(F514=451)∧(F19>1000000)'
      DECIMALS 1
      'HP' DISPLAY SCREEN ADISCLOSURE 406 407 408 409
          U.S. AIRLINES WITH ASSETS OVER $1 BILLION (1981)
-------------------------------------------------------------------
                                |         |         | PRE TAX| PRE TAX
                                | PRE TAX | PRE TAX |  INCOME|  INCOME
                                |  INCOME |  INCOME |      TO|      TO
                                |  TO NET |TO TOTAL |INVESTED|  COMMON
                                |   SALES |  ASSETS | CAPITAL|  EQUITY
-------------------------------------------------------------------
AMERICAN AIRLINES INC             0.9       1.0       2.4       5.1
BRANIFF INTERNATIONAL CORP       (13.2)    (15.5)    (33.7)    165.7
DELTA AIR LINES INC               (0.4)     (0.6)     (1.1)     (1.5)
EASTERN AIR LINES INC             (1.8)     (2.2)     (5.0)    (18.8)
FLYING TIGER LINE INC             (2.6)     (1.9)     (3.4)     (7.5)
NORTHWEST AIRLINES INC             0.4       0.5       0.9       1.0
PAN AMERICAN WORLD AIRWAYS INC    (10.6)    (13.6)    (24.7)    (51.1)
REPUBLIC AIRLINES INC             (3.2)     (4.0)     (6.9)     (63.9)
TEXAS AIR CORP                    (7.2)     (4.0)    (13.4)    (116.8)
TEXAS INTERNATIONAL AIRLINES INC  (5.9)     (3.3)    (13.2)    (196.0)
TIGER INTERNATIONAL INC           (1.9)     (1.1)     (1.8)     (7.2)
TRANS WORLD AIRLINES INC          (0.8)     (1.1)     (2.3)     (5.4)
TRANS WORLD CORP                   1.4       2.1       4.2      11.3
UAL INC                           (2.9)     (3.7)     (9.4)    (13.5)
-------------------------------------------------------------------
SOURCE: DISCLOSURE II DATA BASE
```

Three New Stock Market Data Bases

I.P. Sharp has available three new stock market data bases: United States Stock Market, *Financial Times* Share Information, and *Financial Times* Actuaries Share Indices. These data bases complement other stock market data bases on the SHARP APL Service which include: Canadian Stock Options, *Financial Post* Securities, North American Stock Market, Toronto Stock Exchange 300 Index and Stock Statistics, United States Stock Options, Australian Stock Exchanges Indices, and Sydney Stock Exchange Share Prices.

MAGIC, a software package for data retrieval and analysis, allows you to access, combine, manipulate, report, and plot data from these new data bases, as well as from any of the more than 90 public data bases offered by I.P. Sharp.

And for colour graphics, SUPERPLOT allows you to take data from any data base and produce high-quality colour plots or transparencies.

United States Stock Market

The United States Stock Market data base (USSTOCK) gives you current and historical prices and volumes for securities on all U.S. exchanges: common and preferred stocks, warrants, rights, units, and mutual funds. The data base includes issues traded on the New York (composite), American (composite), Midwest, Boston, Pacific, and Philadelphia-Baltimore-Washington Stock Exchanges, and in Canada, the Montreal and Toronto Stock Exchanges. In addition, the data base includes all NASDAQ issues and some non-NASDAQ over-the-counter issues. Facts include high (ask), low (bid), close, volume, thousand shares outstanding, earnings per share, and dividends per share.

The information is updated daily on the next business day before 09:00 Toronto time. The data starts January 1978. There is no surcharge to access USSTOCK.

Financial Times Share Information

The *Financial Times* Share Information data base (FTSTOCK) provides you with comprehensive information on securities listed on the Stock Exchange in London. It contains not only the share prices published daily in the *Financial Times* of London, but also prices collected but not published. Prices are collected directly from the jobbers at the Stock Exchange by *Financial Times* reporters, allowing coverage of after-hours trading. On a day of heavy trading, this unique coverage of final closing prices can make a significant difference in portfolio and investment appraisals.

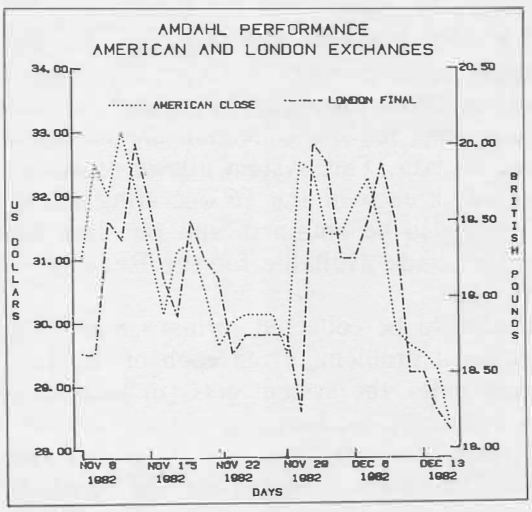
The data base is updated daily, and information dates from 1 June 1982 (or July 1982 for those shares not quoted in the *Financial Times*).

Financial Times Actuaries Share Indices

The *Financial Times* Actuaries Share Indices data base (FTACT) gives index values and associated information for each of the 42 Institute of Actuaries Indices published in the *Financial Times*. The data base is updated daily, commencing 1 June 1982.

Royalty fees for FTSTOCK and FTACT

Royalty fees to access either FTSTOCK or FTACT are one penny per time series item (that is, a single value, for example, a price), to a maximum monthly royalty fee of 400 pounds sterling for combined use of the *Financial Times* data bases. Static facts are not subject to the subscription fee ■



SOURCES: FTSTOCK AND USSTOCK DATA BASES

WIZARDry at Eastman Kodak

The Financial Department of the European Regional Office of Eastman Kodak is responsible for the coordination and control of management and financial data in Europe. This involves the regular collection of various types of data from 16 operating units; the consolidation, analysis, and reporting of that data in London; and, in most cases, transmitting that data to Eastman Kodak headquarters in Rochester, New York.

Over the past few years, the Regional Office has expanded, with the result that the volume and diversity of data have multiplied. Because of this wider range of data, line management and functional specialists are demanding more analyses of data in less time.

To meet these new demands, Eastman Kodak's Financial Department decided that a new approach was needed. This new method:

- Allows easy and fast transfer of data to and from the Regional Office
- Provides data bases for one-off requests, both locally and centrally
- Allows data to be modelled and amended easily.

There were several ways of meeting these requirements and we chose I.P. Sharp. The SHARP APL Service was already being used in-house by Eastman Kodak for other applications. I.P. Sharp's range of powerful packages, their use of APL (Eastman Kodak's approved timesharing language) and the extensive SHARP APL Network met the Financial Department's requirements.

Thus, in 1981, we asked I.P. Sharp to develop a system for the collection of one particular set of data. This system allows details to be entered in each of the 16 operating units, local reports to be obtained, and the data base then to be made available for the Regional Office's use.

The data to be collected forms a multidimensional problem. From each of the 16 operating units, the system gets (in local currency) numerous product aggregation levels and various financial items over quarterly time periods. The values collected are the forecast, revision to forecast, and actual figures. The Regional Office holds the same data with the added dimensions of companies and currency—local currency and U.S. dollars.

To handle this data, we chose I.P. Sharp's software package, WIZARD, as the data base manager. I.P. Sharp consultants wrote additional software to provide further facilities such as customized input for the 16 companies, several standard reports, and conversion from local currency to U.S. dollars. Although the staff of the local companies have not been trained in the use of WIZARD itself, some are now using this powerful package to analyse their own data ad hoc. The Regional Office staff certainly find the ability to consolidate any subset of the data over any dimension or dimensions very useful.

This first system proved to be very successful, and, since then, the Financial Department has added five WIZARD-based systems, with more to come. These all follow the same philosophy and operating procedure as the first one, and thus are quick to set up, and easy for the local companies to use.

WIZARD is also used to manage data created on our computer in a form suitable for online access.

WIZARD is also used extensively in the Marketing Department of the European Regional Office. The data bases are established using the same philosophy as for the Financial Department, but many more features are used. Particularly useful is the ability to plot data from a WIZARD data base using I.P. Sharp's SUPERPLOT package.

Eastman Kodak's experience with WIZARD has been very favourable, and, coupled with the facilities of MAGIC and SUPERPLOT, WIZARD meets our requirements admirably■

Alan Bland, London, Financial Department, Eastman Kodak, European Regional Office

Simplifying Network Messages
Or Who said that?
And what does it mean?

When all goes well, APL timesharing is just the dialogue between you at your terminal and the APL system to which you're connected. But, when problems arise, you get messages. The messages originate at the various waystations between you and APL, or from APL itself. Once you understand where they originate, it's much easier to understand what they mean.

This winter (for our northern hemisphere clients; summer down under) we are introducing some changes in the messages that originate in the SHARP APL Network. This article first explains where messages come from, and then lists and explains the messages.

What is the network?

With the SHARP APL Network, users in different parts in the world can reach APL by making a local phone call. The network consists of a set of minicomputers with at least one minicomputer located in each city where I.P. Sharp has local telephone access. The minicomputers are linked together by cables. Usually the cables are private lines leased from the various telephone authorities.

Each minicomputer is a **node** of the network. A node is connected to at least one other node, and sometimes to several. Each node is able both to communicate with its neighbouring nodes, and also to accept calls from individual terminals.

To link your terminal to a network node, you may dial the number of a phone directly connected to the node. Alternately, your terminal may have a cable going directly to the node (thus being a "hard-wired" terminal). The device at the node which accepts a call is called a **port**. The minicomputer must have one port for each terminal session signed on through it.

Packet switching

One job of the node is to break the message from your terminal into segments. Each segment is called a **packet**. A packet is labelled to show from whom it comes and where it is destined. It also has a sequence number to help reassemble packets in the right order. At the receiving end, the node reconstructs the message from the packets it receives.

Breaking the transmission into packets makes it much easier to verify (and, if necessary, correct or repeat) individual packets. It also makes it much easier for a few long-distance links among the nodes to process the work of many different users, and hence to cut the costs of long-distance communication.

Multiple destinations

The SHARP APL Network serves only users of SHARP APL. But SHARP APL is offered at a number of different computer centres. So when you dial a node of the network, the first thing you must tell the node is *which* SHARP APL system you want to talk to.

Your first transmission includes the character). Following the parenthesis, you may type letters to indicate which SHARP APL system you want. Each node has its own default rules about what to assume if you just type) but don't specify an APL system. (That default rule permits the majority of users of a particular node to reach the system they want without having to name it explicitly.)

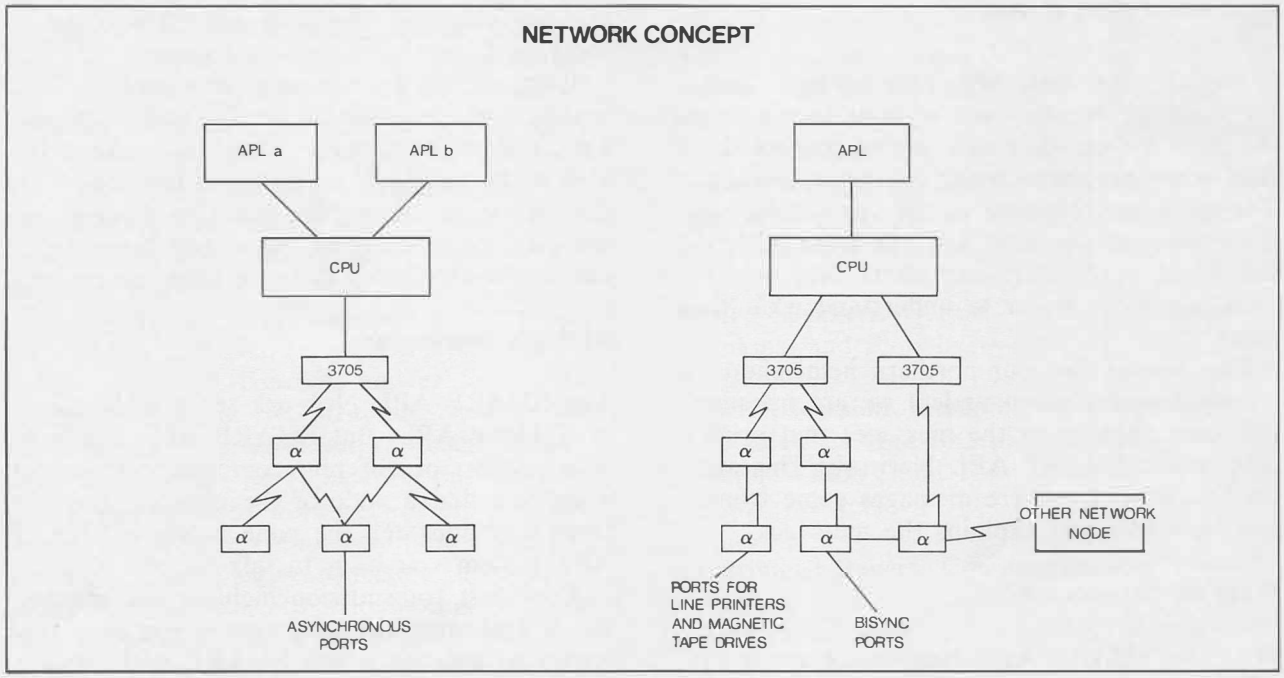
Alphas and 3705s

The minicomputer at each node may be either of two types: a Computer Automation model Alpha LSI 2, or an IBM 3705. These names are universally abbreviated by the people who work with them to "Alpha" and "3705", and this article will do likewise.

Any node, be it an Alpha or a 3705, can both transmit data to the nodes neighbouring it, and handle calls originating from individual users. However, only a 3705 has the additional ability of being able to communicate with the computer (CPU) that runs an APL system.

The network concept diagram doesn't show a particular network, but illustrates the sort of connections that are possible. You can see that the Alphas (α) and 3705s are connected to each other in an arbitrary pattern, except that only a 3705 is connected to an APL system. Some large APL systems may be served by several 3705s.

. . . Simplifying Network Messages



Asynchronous ports are for connection to normal interactive terminals. **Bisync** (binary synchronous) ports are specialized ports for batch mode file transfers.

The chain from you to APL (and back)

The second diagram illustrates the various links that usually exist between you and the APL system you're using. When you first try to connect to APL, you begin by reaching a node on the network. (You may dial the node directly, or, if you link via a public network such as Tymnet, Telenet, or Datapac, you use that network to reach a node of the I.P. Sharp Network.)

The first node you reach must decide what type of terminal you're using, so that it can understand your transmissions and make intelligible replies. It has to identify both the speed of transmission (baud rate) and the data encoding. You must send it the APL character) because that character is sent differently in the various encodings, and thus serves to reveal which one your terminal provides.

Next, the node needs to know which APL system you're trying to reach. If the APL system you ask for cannot be reached, the node sends you the message *NO PATH*. That's a new message; it partially replaces the message *LINES DOWN*.

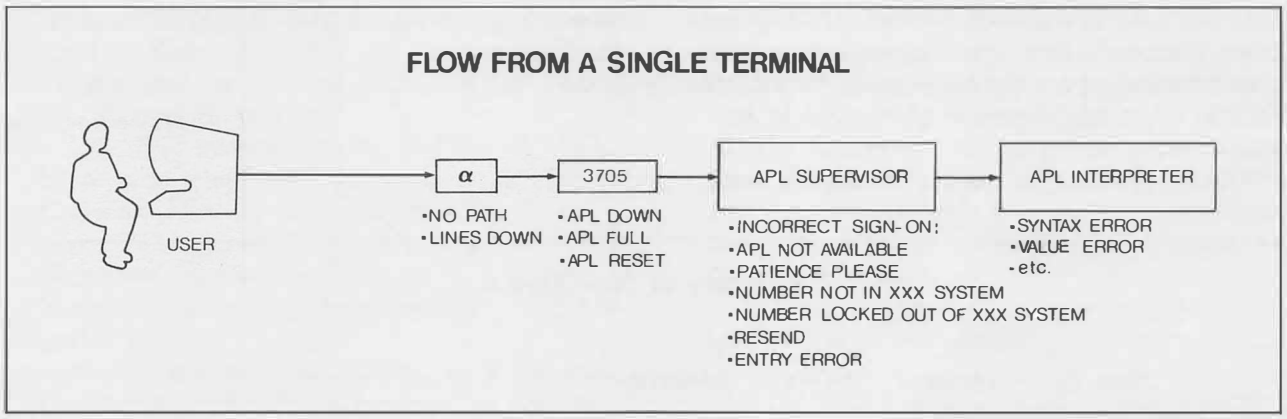
No path may exist for either of two reasons: (1) You've asked for a real APL system, but for the moment there is no way to reach it, perhaps because of an interruption in the link from one node to another. (2) The APL system you've asked for doesn't exist—perhaps because you've mistyped the name. The node you reach doesn't have a way of distinguishing these two situations; it knows only that it can't place a call to the system you've asked for, and so it says *NO PATH*.

A node may also send out the message *LINES DOWN*. With the change, it will do that only if it has successfully connected you to an APL system, but then finds that it has been cut off by a communications failure somewhere between it and APL.

When your local node is able to address a call to the APL system you want, your message is sent to a 3705 serving that CPU. The 3705 passes on to the APL supervisor your request to sign on.

The 3705 may not be able to reach the APL supervisor. In some cases this is because the 3705 itself is swamped and can't accept

... Simplifying Network Messages



more calls. You then get the message *APL FULL*. Or it may be that APL isn't running, in which case you get *APL DOWN*. If you have successfully signed on and subsequently the 3705 notices that APL has been terminated (and has reset its links to the 3705) the 3705 sends you the message *APL RESET*. You are no longer connected to the system.

Messages that originate in the 3705 may be followed by a slash and an abbreviation intended to help the staff diagnose the problem. These needn't concern you (save to note that the infamous *AWAITING FUNCTION FOUR* of former years now appears as *APL DOWN /F4*).

Messages from the APL system itself

You can divide APL into approximately two parts: the **supervisor** that manages the APL service, and the **interpreter** that actually does your work after you've signed on. The following messages are sent by the supervisor. When you see one of these, you know that your call has reached an APL system.

INCORRECT SIGN-ON: means that the first transmission the supervisor receives doesn't match the required format. Your sign-on must be in the form:

)1234567:*PASSWORD*

where 1234567 and *PASSWORD* represent your user number and password respectively. The parenthesis and colon are obligatory; embedded blanks are forbidden. You usually get *INCORRECT SIGN-ON*: when the sign-on includes blanks, the user number includes non-numeric characters, or you've omitted the colon.

APL NOT AVAILABLE is a message that the supervisor sends during part of its start-up procedure. It means that APL is indeed running, but isn't yet ready to accept sign-ons. APL should be ready soon.

PATIENCE PLEASE is sent when you've been using the system and have been disconnected. Possibly your terminal has been disconnected or network problems have prevented your signing off. When you try to sign on again, on the same account number, you receive this message. You will be able to sign on shortly.

NUMBER NOT IN xxx *SYSTEM* arises either because you aren't enrolled at all, or because the password you've supplied is incorrect. (For xxx, substitute the name of the particular system you've reached.)

NUMBER LOCKED OUT OF xxx *SYSTEM* means that the operators have locked your account to prevent anyone's signing on.

There are two messages that the APL supervisor can send both before and after you've signed on. They are *ENTRY ERROR* and *RESEND*. *ENTRY ERROR* arises when the supervisor is unable to decide what APL characters you intended, either because you have sent TAB characters without defining TAB positions,

. . . Simplifying Network Messages

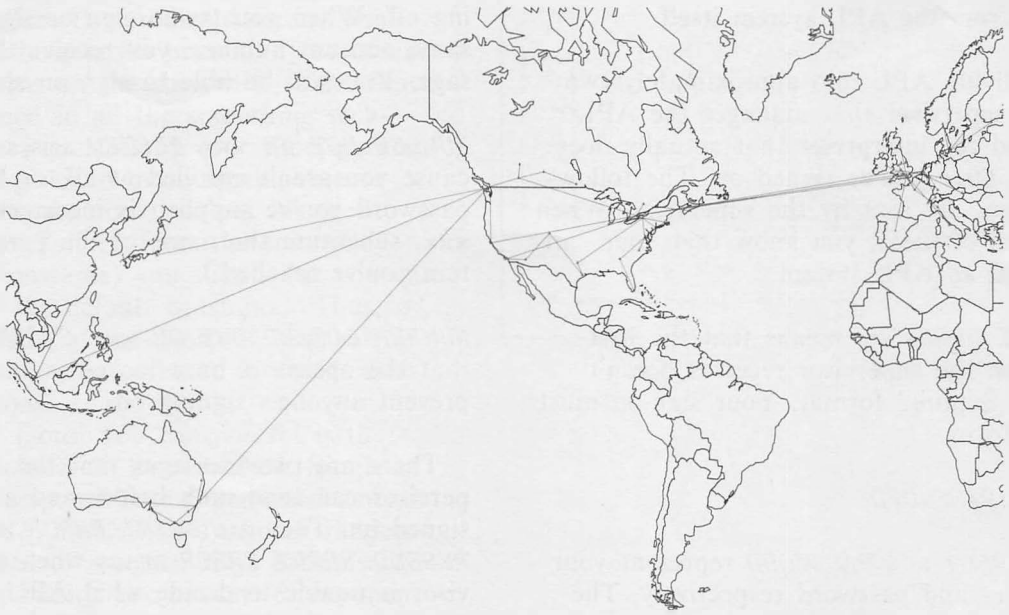
or because you have constructed overstrikes that don't form valid characters. *RESEND* indicates a transmission that the supervisor is unable to handle, usually because of its extreme length, or, rarely, because of a problem in transmitting it.

Once your call has passed through a node, a 3705, and the APL supervisor, and you've signed on to an APL system, at last you're free to make APL mistakes, and receive messages such as *VALUE ERROR*, *SYNTAX ERROR*, and so on ■

Paul Berry, Palo Alto

Summary of New Messages

New Error Message	*Origin*	*Meaning*
NO PATH	Alpha	SHARP APL system cannot be reached.
LINES DOWN	Alpha	After you are successfully connected to an APL system, there is a communications failure between the node and APL.
APL DOWN	3705	APL isn't running on the host computer that the 3705 is connected to.
APL FULL	3705	The 3705 cannot accept any more calls.
APL RESET	3705	You are successfully connected to an APL system, but APL has been terminated.



I.P. Sharp Communications Network

JANUARY / FEBRUARY 1983
Volume 11 Number 1

I.P. Sharp newsletter

technical supplement 42

Searching, Part V

This series of articles has presented a number of techniques for searching in APL. Methods for searching lists, tables, texts, and files have been explored. In this final article of the series, we present methods for searching based on sound, meaning, and spelling.

5) Searching by sound, meaning, and spelling

Trying to use computers to handle natural languages in any sort of way that approaches human use of the same language is a humbling experience. Why? Because reading, speaking, and listening, which come to us effortlessly, are really very complicated processes. Consequently, our goals here are quite limited.

What exactly are we going to present in this article? We want to search lists of words, based upon sound, meaning, and spelling. Like APL variables, words have names (spelling) and referents (meaning). In addition, since language is spoken, words have sound as well. Unfortunately, there is not a one-to-one relationship between these attributes. Two words may have the same name (and sound), but different meanings. These are **homonyms**. There are also words which have different names (and sounds), but the same meaning. These are **synonyms**. Other words have the same name, different meanings, and maybe different sounds (e.g. "lead" pipe *vs* "lead" singer). We call these **homographs**. Lastly, there are words which have the same sound, different meanings, and maybe different names. These are **homophones**. We can use all of these concepts to make APL application systems more tolerant of the variation in human behavior.

Searching by sound

When we enter words at a keyboard, we aren't always copying written text. A good example is a telephone operator answering a "Directory Assistance" request. Such a person will spell the word based on how it sounds. In addition, language is an oral/aural skill. So, we often remember words the way they sound, even if we have seen them written out.

If someone keys in a word we can't find after searching by traditional methods, we may want to search for a word which sounds like the one entered. If so, we can use the methods described below.

We need to create a function which takes two lists of words as arguments. The left argument is the list to be searched in. The right argument is the list to be searched for. The result is like that of dyadic iota—the first index where a match is found, or ⎕*IO*+ the number of words in the left argument.

```
SOUNDSEARCH1◇ (WORDENCODE SOUNDENCODE α
  )⍳WORDENCODE SOUNDENCODE ω
```

If the same list is to be searched repeatedly, it should be encoded once and used with a function like the following:

```
SOUNDSEARCH2◇ α⍳WORDENCODE SOUNDENCODE ω
```

Now we must consider how to represent the sound of a word. There are significant regional differences in the pronunciation of vowels, even among people who speak the same language. Just get a native of Texas, North Carolina, or Boston to pronounce "Texas", "North Carolina", or "Harvard Yard" respectively, and you will see (*oops, hear*) the problem. Because the sound of a vowel depends so greatly upon the accent of the speaker, it may not be meaning-

. . . Searching, Part V

ful to distinguish vowel sounds. We can convert all vowels to a single sound. Or, we can eliminate them altogether.

Many consonants sound similar to other consonants. We encounter a problem when we ask the question: to whom do they sound similar? In addition, the same consonant does not always sound the same. The answer to these problems is to encode the same word several ways. This requires searching different representations of the same list, but proves to be much more effective.

Three algorithms for reducing a literal vector of words to sound representations are listed below. All of them remove vowels. They all also reduce duplicate adjacent sounds to a single letter. They all map similar sounding consonants into a single letter. The difference between the algorithms depends upon which consonants they treat as having the same sound. In addition, the last algorithm also handles consonant pairs which create a single sound as single units.

```
    ∇ SOUND←SOUNDENCODE1 WORD
[1]  WORD←(~WORD∈'AEIOUYH')/WORD
[2]  WORD←' BBBBBRRMMDDCCCCCCGG'[
           ' BPFVWRLMNDTCKQSXZGJ'⍳WORD]
[3]  SOUND←(¯1↓1,WORD≠1⌽WORD)/WORD ∇

    ∇ SOUND←SOUNDENCODE2 WORD
[1]  WORD←(~WORD∈'AEIOUHWY')/WORD
[2]  WORD←' CCCCCCCCLRBBBBDDMM'[
           ' CGJKQSXZLRBFPVDTMN'⍳WORD]
[3]  SOUND←(¯1↓1,WORD≠1⌽WORD)/WORD ∇

    ∇ SOUND←SOUNDENCODE3 WORD
[1]  WORD←(~WORD∈'AEIOUY')/WORD
[2]  WORD←(~(¯1↓0,WORD∈'WSPCT')∧
     WORD='H')/WORD
[3]  WORD←(~(WORD∈'ND')∧1↓(WORD='G')
     ,0)/WORD
[4]  WORD←(~(WORD∈'P')∧1↓(WORD='S')
     ,0)/WORD
[5]  WORD←' HHSPPPPCCCCCCDDDLLMM'[
           ' HWSVFPBQCJGKXDTZLRMN'⍳WORD]
[6]  SOUND←(¯1↓1,WORD≠1⌽WORD)/WORD ∇
```

The first algorithm is used by the U.S. Immigration and Naturalization Service to classify immigrants according to the sound of their names [1]. The second algorithm is the "Soundex" procedure . The third is based upon the classification of sounds by the physical actions necessary to make them. None of the algorithms does a perfect job of representing sounds.

We can see, however, that they do handle simple variation quite nicely.

```
   SOUNDENCODE1 'BERRY BURY BARRY BARI'
BR BR BR BR
```

Some words are harder.

```
   WORDS←'BROWN BRAUN BRAWN BRON'
   SOUNDENCODE1 WORDS.
BRBM BRM BRBM BRM
   SOUNDENCODE2 WORDS
BRM BRM BRM BRM
   SOUNDENCODE3 WORDS
PLHM PLM PLHM PLM
```

The fact that the one algorithm does better on this particular word than others shows why searching several representations of the same list is a good way to do this type of searching.

Once we have reduced the words to their essential sounds, we must encode them to do the actual search. We can use algorithms described in previous articles from this series for this purpose. *ENCRYPTCHAR* can convert the words into a single number, or *ENCLWORDS* can enclose each word as a separate array.

Searching by meaning

Searching by sound provides for toleration of minor differences in the way people spell words. We can extend this concept further and allow them to enter different words entirely. We can do this by treating certain words as if they had the same meaning.

If we were building a program to maintain a data base, we might have three commands: ADD, CHANGE and DELETE. We might also want to allow users to enter synonyms for these commands.

ADD—APPEND
CHANGE—MODIFY, ALTER
DELETE—REMOVE

If the workspace were an "open system" in which you typed function names in immediate

. . . Searching, Part V

execution mode, you could simply provide cover functions for the one which did the actual work.

```
     ∇MODIFY
[1]  CHANGE ∇
     ∇ALTER
[1]  CHANGE ∇
     ∇CHANGE
[1]  ⍝ THIS FUNCTION DOES THE WORK ...
```

If the workspace is a "closed system" in which you answer prompts from the program, you need a way to map the user's entry to a valid command. How can we represent a list of synonyms as a data structure? Conceptually, what we want is a vector of vectors, which looks like the following:

```
('ADD','APPEND'),
('CHANGE','MODIFY','ALTER'),
('DELETE','REMOVE')
```

If we encode these words as integers, we can represent the beginning of a subvector by multiplying it by ¯1.

```
¯22707 625614786,
¯1277864964 525007922 21123990,
¯1627916913 7052533488
```

Given this type of structure, the following program will search for and return synonyms.

```
   ∇ SYNONYMS←LIST MEANINGSEARCH1 WORDS
     ;⎕CT
[1]  ⎕CT←0
[2]  SYNONYMS←((|LIST[⌈\(LIST<0)×
     ⍳⍴LIST]),¯1)[(|LIST)⍳WORDS] ∇

   W←ENCODEWORDS 'ADD ALTER NONSENSE'
   SYNONYMS MEANINGSEARCH1 W
22707 1277864964 ¯1
```

If you are using SHARP APL, you can also use enclosed arrays to handle synonyms. The list is represented by a vector of enclosed vectors. The first element of each vector contains the enclosed version of the principal word. Any following elements contain the enclosed synonyms.

```
   ⎕PS←¯1 ¯1 ¯2 ¯2
   ⎕←3 1⍴SYNONYMS

┌──────────────┐
│┌───┐┌──────┐│
│└┌─┐┘└──────┘│
│ │A│ │APPEND│ │
│ └─┘ └──────┘ │
└──────────────┘
┌──────────────────────┐
│┌──────┐┌──────┐┌─────┐│
│└┌────┐┘└┌────┐┘└┌───┐┘│
│ │CHAN│  │MODI│  │ALT│ │
│ └────┘  └────┘  └───┘ │
└──────────────────────┘
┌────────────────┐
│┌──────┐┌──────┐│
│└┌────┐┘└┌────┐┘│
│ │DELE│  │REMO│ │
│ └────┘  └────┘ │
└────────────────┘
```

If you have such a data structure, the following program will search for and return synonyms.

```
   ∇ SYNONYMS←LIST MEANINGSEARCH2 WORDS
[1]  SYNONYMS←((,1↑⍤0>LIST),<'')
     [⎕IO+.≠∧⍀~(<WORDS)∈⍤0>LIST] ∇

   W←ENCLWORDS ' ADD ALTER NONSENSE'
   ⎕←T←SYNONYMS MEANINGSEARCH2 W
┌──┬──────┬┐
│A │CHANGE│⍬│
└──┴──────┴┘
   ,⍤0>T
3 6 0
```

The essence of this function is the derived function produced by ∈⍤0>. It goes through each synonym list in turn, searching for a match with the right argument. This implied loop is the real power of the ON operator. You can see how it works by examining the three expressions which follow. They are equivalent for the arguments we have used here.

```
(<WORDS)∈⍤0>LIST

((><WORDS)∈>LIST[1]),
((><WORDS)∈>LIST[2]),
((><WORDS)∈>LIST[3])

(WORDS∈>LIST[1]),
(WORDS∈>LIST[2]),
(WORDS∈>LIST[3])
```

Note that the result of this expression will be a matrix. It will have as many rows as there are elements in *WORDS*, and as many columns as elements in *LIST*. This is because ON creates a new first dimension (here rows) and stacks the results on top of each other as it goes through the loop ■

Robert Metzger, Rochester

Editor's note: This article on searching by sound, meaning, and spelling will be continued in *Technical Supplement 43.*

The Tree Report
Or Scanners do not Live in Vain

In *Technical Supplement 40*, I introduced the topic of trees by showing how to draw them. Now, you might think that APL trees are good for pie or cider, but useless for making your daily bread. In this instalment I'm going to prove you wrong by showing a simple and efficient way to generate and format a commonly used style of business report, using an extension of plus scan to trees. This will lead me into the subject of how to represent trees efficiently in the workspace.

But first, let's define the practical problem to be solved. In a typical hierarchical organization, items of cost or revenue are grouped into sets, sets of sets, and so on. Examples are easy to think of: a company split into divisions, each composed of departments; or a multinational firm which groups its business by national subsidiaries, their regional groups, and local branches. Whatever the names, the structure is broadly the same. The problem, then, is given the basic data at the leaves of this tree (individual cost items, or revenues by branch), can we find a simple way of subtotalling them by department, then by division, and so on? If you think about the problem in terms of branches of the tree structure which describes the organization, we have values at some of the nodes, usually the leaves, and we want to find the running sum—the plus scan—going down the branches of the tree.

Well, the way to get to the root of this problem is to sort it out by meditating upon the true path(s). No, that wasn't a promotion for mysticism. Remember that in *Technical Supplement 40*, I defined the **path vector** to a given node as being the set of directions to follow in going from the root of the tree to the node. When we go up from a node on the I-1 level, our path vector says to take the *PATH*[*I*]th branch. If we list out the path vector from the root to each node in the tree, the set of all such paths can be thought of as a path matrix *PATHS*. As I sit here with my stomach grumbling about the dryness of this article, it occurs to me we need a juicier example.

Figure 1

```
INDENT←2×1-LEVEL←+/'X'≠PATHS
(INDENT⌽NAMES),PATHS

STRUDEL CORP.        1XXX
  PASTRY DIV.        11XX
    BATTER DEPT.     111X
      FLOUR          1111
      SUGAR          1112
      YEAST          1113
    CUPCAKE DEPT.    112X
      SYNTHOMIX      1121
      PSEUDOICING    1122
  FILLING DIV.       12XX
    APL FILLINGS     121X
      APL TREES      1211
      FRUIT          1212
      TIMESHARING    1213
    CREAM FILLINGS   122X
      CLOTTED MOO    1221
      STICKY GOO     1222
      PROGRAM FOO    1223
```

Note that in the path matrix I've used an X to pad out all the paths to the same length. I've listed the names of the different items in a rather special order, and indented them appropriately, so we can see the tree structure easily. However, we don't have to assume the data is given to us in this order, for once we have the path matrix, we can sort things into the right order. How? Imagine that I replaced each X on the matrix with a 0 . . . then the rows are numbers, listed (in figure 1) in increasing order. So if we are given another path matrix *P* in any order, *S*←'X12345'⍋*P* gives us a sort vector from which we can get the ordering of figure 1; that is *PATHS*←*P*[*S*;].

As the next step, let's define our other variables. Our item names will be in a character matrix called *NAMES* which is left-justified. And our data is in a vector *V* in the same order as the names and path matrices. Initially, of course, *V* simply has zeros in the positions corresponding to totals. The only positions having values initially are at the leaves of the tree. Now we want to do our plus scan down the branches.

We can do it by generalizing the "partitioned plus scan" function—a favourite of many APLers. Suppose we want to sum each

section in a partitioned vector separately, without resorting to looping. We need information about where the partitions are in the vector—one way to describe them is by the list *PL* of partition lengths.

```
     ∇R←PL PPLUSSCAN V
[1]  R←0,+\V
[2]  R←(1↓R)-PL/R[¯1↓⎕IO+0,+\PL]∇
```

Note that in line [2], the last value in *R* preceding the current partition is subtracted to remove the contribution to the scan of all predecessors to the partition. So the essence of the procedure is to do one plus scan to the whole vector, and then to subtract out values at "predecessor" positions.

To do this with a tree, we want to be sure that points lower in the tree—subtotals, totals—always *follow* the data items belonging to them in the scan. (*Why?*) Clearly that requires a different order from the kind shown in figure 1. In fact, the order we want is uniquely determined, and is shown in figure 2.

Figure 2

```
(INDENTΦNAMES[S;]),PATHS[S;]
      FLOUR          1111
      SUGAR          1112
      YEAST          1113
   BATTER DEPT.      111X
      SYNTHOMIX      1121
      PSEUDOICING    1122
   CUPCAKE DEPT.     112X
 PASTRY DIV.         11XX
      APL TREES      1211
      FRUIT          1212
      TIMESHARING    1213
   APL FILLINGS      121X
      CLOTTED MOO    1221
      STICKY GOO     1222
      PROGRAM FOO    1223
   CREAM FILLINGS    122X
 FILLING DIV.        12XX
STRUDEL CORP.        1XXX
```

How do we get the tree into this new order? Try substituting 9 for X in the path matrix, and look at the numbers formed . . . once again they're in increasing order. So *S*←'12345*X*'⍋*PATHS* gives it to us. Call this order "Left-High" order, and the one in figure 1, "Left-Low" order. The "Left" reminds us that the path vectors are left-justified in the path matrix, while the "High" or "Low" tells us whether X comes after or before the digits in the sort order. From here on, I'll assume all our variables (*NAMES*, *V*, etc.) have been sorted into L-H order.

Diligent devotees of APL will recognize that what I call Left-Low order corresponds to Kenneth Iverson's "left list" for trees, as defined in his 1962 book *A Programming Language*. I'll return to this tree ordering topic in another article . . . I'm too disappointed at having been scooped by 20 years to go on with it now!

Now, how do we find where the logical "predecessor" is for a given item in an L-H (Left-High) tree?

```
PRED←(⍋'X12345'⍋PATHS)-LEVEL
```

LEVEL is just the familiar height of each item in the tree. If you want to understand how this works, a picture is worth many words: stare at figure 3 for a while. (And don't get discouraged right away, it only took me a year to figure this out.)

Figure 3

```
LEFT-HIGH LIST       L    P   L+P  INDENTΦNAMES[⍋L+P;]
      FLOUR          4    0    4  STRUDEL CORP.
      SUGAR          4    1    5    PASTRY DIV.
      YEAST          4    2    6      BATTER DEPT.
   BATTER DEPT.      3    0    3        FLOUR
      SYNTHOMIX      4    4    8        SUGAR
      PSEUDOICING    4    5    9        YEAST
   CUPCAKE DEPT.     3    4    7      CUPCAKE DEPT.
 PASTRY DIV.         2    0    2        SYNTHOMIX
      APL TREES      4    8   12        PSEUDOICING
      FRUIT          4    9   13    FILLING DIV.
      TIMESHARING    4   10   14      APL FILLINGS
   APL FILLINGS      3    8   11        APL TREES
      CLOTTED MOO    4   12   16        FRUIT
      STICKY GOO     4   13   17        TIMESHARING
      PROGRAM FOO    4   14   18      CREAM FILLINGS
   CREAM FILLINGS    3   12   15        CLOTTED MOO
 FILLING DIV.        2    8   10        STICKY GOO
STRUDEL CORP.        1    0    1        PROGRAM FOO
```

Now that we have *PRED*, doing all our subtotalling in one fell swoop is easy.

. . . The Tree Report

```
    ∇ V←PRED TREESUMLH V
[1]  ⍝DOES +\ DOWN BRANCHES OF TREE WITH NODE VALS V IN L-H ORDER.
[2]  V←↓\V←,V
[3]  V←V-(0,V)[⎕IO+PRED]
    ∇
```

So, let's write the report. We'll improve its format by indenting item names by their height in the tree, and putting skips after total lines.

```
SUMS←PRED TREESUMLH V⋄I←INDENT
FS←'20A1,BCP<$>I8,1A1'
FS ⎕FMT (I⍱NAMES;SUMS;(I≠1↓I,0)\CR)
```

```
        FLOUR            $3,140
        SUGAR            $2,508
        YEAST            $1,270

    BATTER DEPT.         $6,918

        SYNTHOMIX          $350
        PSEUDOICING        $420

    CUPCAKE DEPT.          $770

  PASTRY DIV.            $7,688

        APL TREES          $123
        FRUIT              $321
        TIMESHARING     $12,345

    APL FILLINGS        $12,789

        CLOTTED MOO      $2,590
        STICKY GOO         $850
        PROGRAM FOO        $200

    CREAM FILLINGS       $3,640

  FILLING DIV.          $16,429

STRUDEL CORP.           $24,117
```

As you can see, the whole thing is as simple as APL pie. Of course this is only a sketch of how you would use the algorithm in an actual report function, but the basic ideas are here. I hope at any rate that I've given you something to think about. But if you found this too easy, I've some more problems to occupy you until next time. Good luck!

Up the Tree without a (L)adder

I showed you how to do plus scan going *down* the branches of a tree in L-H order. Can you write a program to do a plus scan going *up* the branches? This has useful applications in doing project analysis or decision tree studies, where you want to compare the sum of times or costs along various paths. *Hint*: there are two different solutions, each making use of a different tree ordering. One is a *L*ooping *adder*, or ladder, which can be written extremely simply, with the right tree order. But how can you go up the tree without a ladder? I found a solution that doesn't loop . . . can you?

Lost among Pathless Trees

You can't see the forest for the trees, for you've lost your path matrices!! You must keep a *level* head, for you still have the level vector *L* for a tree which you know is in L-H order. Can you write simple, non-looping, non-recursive one line programs to do the following???

A. Given *L*, return *PRED*, the predecessor vector used by *TREESUMLH*.
B. Given *L*, return *PARENTS*, a vector which for each node in the tree gives the position of the parent of that node in the L-H ordered tree.
C. Given *L*, return *RLPOS*, a sort vector such that *PATHS*[*RLPOS*;] is in R-L order (Iverson's "right list").
D. Given *L*, return the path matrix *PATHS* ■

Clement Kent, Toronto

Additional Access in Germany

A direct link has been commissioned between the I.P. Sharp Network and the Datex-P service operated by the Bundespost in the Federal Republic of Germany.

This link provides access to SHARP APL by a local telephone call in 17 major cities: Augsburg, Berlin, Bielefeld, Bremen, Dortmund, Düsseldorf, Essen, Frankfurt, Hamburg, Hannover, Karlsruhe, Cologne, Mannheim, Munich, Nuremburg, Saarbrücken and Stuttgart.

Asynchronous terminal users may connect to Datex-P at either 30 characters per second (cps) or 120 cps. At 120 cps, the modems used may be either 1200 bits per second both ways, or an asymmetric 1200/75 bits per second. Modems conform to the international CCITT standard recommendations (V series).

To find out the access telephone numbers, load the workspace `1 NETWORK`, and type `PHONES`. For a description of sign-on procedures, type `DATEXHOW` in the same workspace ■

New Dial Access Numbers

Brussels

30 cps 02 648 5995
120 cps 02 648 7460

Düsseldorf

30 cps 0211 450681

Paris

30 cps 225 83 60
120 cps 225 86 50

New Functions in MAGIC

New statistical and forecasting functions have been added to MAGIC. These functions extend the curve fitting, moving average, and exponential smoothing techniques already available. All the functions are compatible with MAGIC's convenient tabulation and plotting capabilities, and therefore provide you with a comprehensive way to retrieve, manipulate, forecast, and display your data.

Curve fitting

The function `CURVEFIT` can fit and forecast a series based on eight types of curves: linear, quadratic, cubic, exponential, log time, log-log, inverse, and S-shaped. Curve fitting techniques are appropriate for long-term forecasting. First, you choose the type of curve appropriate to your data. This might be linear, for a constant trend; exponential, for a constant growth; or S-shaped, for the life cycle of a product. `CURVEFIT` then calculates the parameters of the curve that are the statistical "best fit" for your data, and uses these to calculate predicted values.

Moving averages

Moving averages are applied to a time series to remove randomness from the series. By taking successive averages of sets of data, an averaged series is produced in which the random fluctuations (or "noise") have cancelled each other out.

The number of data points to be averaged affects the amount of randomness eliminated from the series. Depending on this choice, you can eliminate the effects of trend or seasonality.

For example, a 12-point single moving average, applied to monthly data for a highly seasonal commodity, such as ski sales or airline traffic, eliminates seasonality. Because each value in the resultant series is an average of one year's data, monthly fluctuations are smoothed out.

The single moving average is only one of the moving average techniques available when you use the function `MOVAVG`. The centred moving average, useful when you choose an

even number of points to be averaged, produces an average series centred within the time period.

Spencer's 15-term, and Henderson's 9-, 13-, and 23-term weighted moving averages are appropriate for isolating the trend-cycle component from monthly data which has already been seasonally adjusted. Henderson's 5-term moving average performs the same function for quarterly data.

The 3×3, 3×5, and 3×9 double moving averages are useful for smoothing data over a particular period (e.g. all Januaries) to remove randomness before estimating seasonal factors.

Exponential smoothing

An alternate method for removing randomness from a series is exponential smoothing. Exponential smoothing techniques smooth and forecast a time series, based on averaging over *all* past observations. The smoothed values are calculated as a weighted function of past observations, a function which is, in effect, an exponentially weighted moving average. Forecasts based on exponential smoothing are appropriate for immediate or short-term predictions.

Using the exponential smoothing techniques available in MAGIC you have control over two factors that affect the predicted series. The first factor is the relative impact of recent and past observations on the smoothed value. Since the weights used in all the smoothing techniques decline exponentially over time, greater importance is given to more recent observations. However, the actual values for the weights are determined from the smoothing constant (or constants) you supply (or you can choose to use the default constants). Thus, by changing the size of the constant, you can modify the relative importance of recent and past observations.

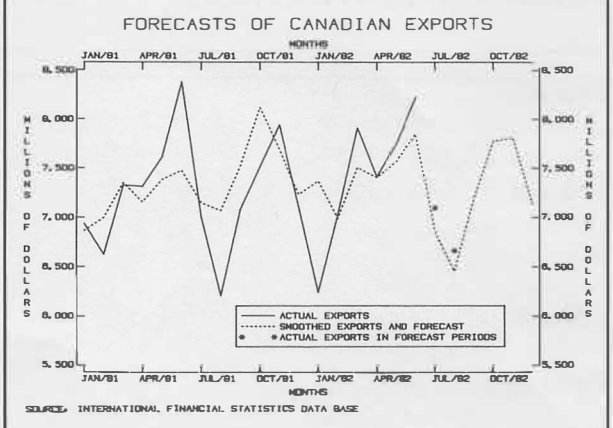
The second factor you can control is the type of randomness to be smoothed. Single and adaptive response rate exponential smoothing (*SMOOTHSIN*, *SMOOTHARR*) smooth randomness in the level of a series only. Brown's one-parameter and Holt's two-parameter exponential smoothing techniques (*SMOOTHBOP*, *SMOOTHHOL*) smooth randomness in both the level and the trend. And Holt-Winter's method and Winter's multiplicative exponential smoothing method (*SMOOTHHW*, *SMOOTHWIN*) smooth randomness in the level, the trend, and the seasonal components of a series.

Example

In the following example, Canadian export data from the International Financial Statistics data base is retrieved for the period January 1978 to June 1982. The data is then smoothed using Holt-Winter's smoothing method, and forecast ahead six periods. The accompanying SUPERPLOT shows how closely the forecast values match the actual data for July and August 1982 ■

Caroline Haythornthwaite, Toronto

```
CLEAR ◊ NOYEAREND ◊ NOZERO
SCALE 1000000
MONTHLY,DATED 1 78 TO 6 82
1 PUT IFS 'CAN/70'
EXTEND TO 12 82
2 PUT 0.2 0.2 0.5 SMOOTHHW ITEM 1
3 PUT (IFS 'CAN/70'), ONLY AT 7 82, 8 82
PLOTSET     ⍝SET SUPERPLOT OPTIONS
'□' PLOT ABOVE, ONLY 1 81 TO 12 82
```



Computer Applications Dynamics

Life is flux. Everything is in a constant state of change. The only thing that remains static is the fact that the universe is in a perpetual state of dynamic evolution. Despite all this, people write computer systems and expect them to last a lifetime. In reality, computer systems tend to be short-lived creatures. They are continually modified, enhanced, and groomed to remove bugs.

Computer systems are never a hundred per cent perfect. They can always be improved. Some of the reasons for this are:

- With regard to system specification, it is impossible for a meagre, rationally thinking programmer to comprehend fully the subtle nuances of abstract thought that users employ to describe their application. Often, users think by arbitrary rules of thumb, gut reaction, and intuition. These don't easily translate into flowchart symbols or program logic.
- Once a computer system has been written, the programmers who wrote it are never happy with it. When an application is rewritten, invariably it is improved. In many cases, an application needs to be written two or three times before an elegant and efficient design is achieved.
- Users are never satisfied with any system. Once they start to use a system, they think of all the little features they should have asked for in the first place. Also, computer systems can radically affect the job of the user by solving a set of problems. At the same time they highlight the need for solutions to other problems. Consequently, users have a voracious appetite for new enhancements and features.

All this change is bad news for the data processing staff and their ulcers. For this reason, they often develop very thick skins, and tend to say "impossible", "can't be done", or "I quit", when asked about improvements to systems. To get around possible changes to specifications, DP departments often resort to the "human wave" approach, where a team of system analysts crawl all over the user's department, and an army of programmers work flat out for two years, writing a megasystem with every conceivable bell and whistle the user can possibly ask for.

The other method of dealing with change is to allow for it. Here, the programmer is an essential part of the ongoing use of the system. Just as the system responds to various commands, the programmer responds to demands made upon him to adapt the system. This *adaptive system* approach allows systems to be modified and extended easily and quickly. Programming in many languages is very time-consuming and tedious. Consequently, programmers are frequently fighting a losing battle in their rearguard action against the ravages of time. For this reason, truly adaptive systems can only be written in a very high-level language.

All systems have a finite useful working life. System longevity depends on two things: the *volatility* of the application, and the *adaptability* of the system. Applications range from one-off models to systems that run for years without change. Stable, well understood systems can be made adaptable by building many options into the system as in the "bell and whistle" approach. The more volatile the system, the more the user needs to become involved with the software. In static systems, the computer language is transparent to the user. With volatile systems, the adaptability of the code and the productivity of programming changes is paramount.

In order to make a system adaptable, it has to be simple, well structured, readable, and concise. All too often, a system is a sprawling mass of sloppy code with a patchwork of fixes and alterations. As a system becomes more and more untidy, the maintenance burden rises until it becomes more difficult to modify a system than to rewrite it. At this point the system has a bad attack of "future shock", and a new version of the system is developed from the ashes of the old.

Bottom-up approach

A system has to evolve or die. To prevent a system from becoming a dinosaur, it needs to be structured so that after each reincarnation of the system, a sufficient amount of the old system must survive for us to recognize it as being the same system as before. In general,

the utilities in a system live much longer than any single incarnation of the system. Compact, useful subroutines, or functions, are more likely to survive the trauma of system revisions. In order to increase the percentage of code that will remain intact from one version to the next, the system must be structured as small, independent, and reliable units of code. To achieve this, a *bottom-up* approach to system design is essential.

Most systems have a *version 1* written using a *top-down* approach. Here, the system is built from simple concepts and explodes them into a hierarchical tree structure of other simple concepts. The top-down approach works from the core of the application outwards, and propagates outward working with a small set of ideas at a time. What results is a nicely structured, but massive collection of code. When it comes time to write *version 2* of the system, there is always the opportunity to simplify the code, locating common patterns within the code, and making these separate subfunctions. This process of *factorization* is an iterative process. Each simplification reduces the complexity and bulk of the system, which in turn makes it easier to spot further areas of simplification. To try to simplify *version 1* totally in one go is almost impossible, and can lead to a state of utter confusion, known as "cerebral meltdown". The factorization of code works from the bottom upwards, starting from the utilities and progressing through the level of system modules to the complete system.

The worst thing that can happen to a system, in terms of its adaptability, is that each modification to the system increases its complexity. If the opportunity to simplify the complexity of the code is not grasped, then it condemns the system to a short and unhappy life. If an application is going to be used for a long time, then it's a wise investment of time to make it adaptable ■

Peter Airs, Toronto

Register Now for APL83

APL83, an international APL conference, will be held in Washington, D.C., on 10 April to 13 April 1983. Sponsored by the Association for Computing Machinery (ACM) and its Special Interest Group on APL (SIGAPL), the conference features:

Invited Speakers: Gerald Weinberg, Michael Jenkins, and Paul Berry

Panel Discussions (moderators): Large Scale Applications (Ian Sharp), Microcomputers (John Wilson), and New Directions in APL (Garth Foster)

Contributed Papers: Commercial and Scientific Applications, Language and System Features, Application Development Tools, Micros, and Teaching Methods

Tutorials: Introduction to APL, Writing APL Applications, General Arrays, Advanced APL Programming Techniques, Comparison of Other Languages to APL, and APL Data Structures and File Design

Exhibits and Product Forums: North American and European companies will exhibit the latest in timesharing services, application products, in-house software and hardware, microcomputers, terminals, educational supplies, and other APL products and services.

Registration fees before 1 March are $135 U.S. (member), $160 U.S. (non-member), and $45 U.S. (student).
For more information, please contact:

Janet Cramer, APL83 Registrar and Treasurer
c/o I.P. Sharp Associates
2033 K St., N.W., Suite 305
Washington, D.C. 20006

Telephone: 202 293-2915

MAILBOX code: *JIC* ■

I.P. Sharp Associates (Hong Kong) Limited

雅博深(香港)有限公司

I.P. Sharp Associates (Hong Kong) Limited has recently translated its name into Chinese. Although both English and Chinese are official languages in Hong Kong (for legal purposes, English only), the majority of the population speak Chinese only. A name which can be easily read and pronounced in Chinese is, therefore, fitting and appreciated by the Chinese, particularly since not all Western companies bother with translations these days.

After much consideration we have settled on a Chinese name with three characters. Their pronounciation is similar to "I.P. Sharp". These three characters are:

雅 博 深

Elegant	Wide spectrum (of knowledge)	In depth (knowledge)

Each character has its own individual translation. Combined together, additional meanings are implied. The last two characters of the name remind one of the words of the highly respected scholar, Hu Shi Chi: "One's knowledge should be like a pyramid—tall and with a wide base." The words "wide base" and "tall" correspond to the second and third characters of the name respectively. Together, all three characters could constitute a description of the Chinese view of a well-educated person, a description that all our staff try to live up to■

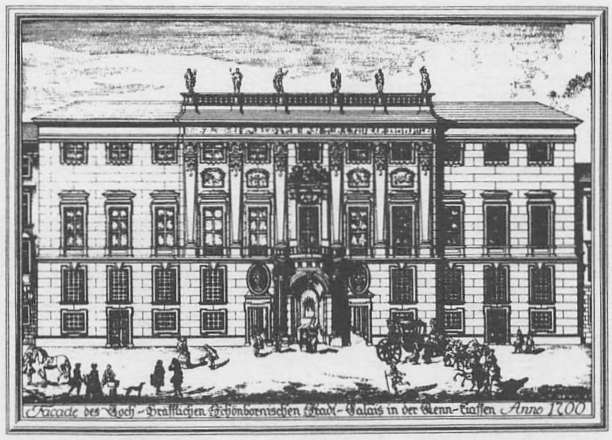
Philip Lai, Hong Kong

Vienna

Our modern telecommunications equipment provides a vivid contrast to the historic building which is the new site of the I.P. Sharp office in Vienna. The office is in the Palais Schönborn Batthyany, built in 1700 by Fischer von Erlach. He was also the architect of the beautiful, baroque St. Charles church (Karlskirche) in Vienna.

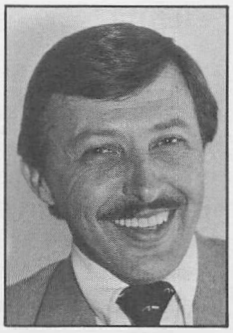
One of the advantages of this beautiful location is the natural environmental control provided by the Palais' walls, which are 125 cm (49.2 inches) thick, and the ceilings, which are 4.5 m (14.8 feet) high. It is cool in the summer and warm in the winter.

The other advantage is that the office is in the centre of the city, close to all the major banks. Please drop in for a visit if you are in the area■



Today, the Palais Schönborn Batthyany looks the same as depicted in this 17th century etching. The horses were replaced, unfortunately, by ugly cars.

Rochester



Fletcher McTaggart has been appointed international software sales manager. In this newly created position, he is directing the marketing of system and applications software for our in-house installations worldwide.

Prior to this appointment, Fletcher was general manager of SHARP APL timesharing in the United States since 1976. Under his management, timesharing revenues in the United States increased by a factor of ten.

Fletcher brings to this position a thorough understanding of the needs of our customers, as his past experience shows. I.P. Sharp's first U.S. employee, he opened and was manager of the Rochester office. Later he became regional manager. Before joining I.P. Sharp, he was a systems analyst with Xerox Corporation, a programmer with Eastman Kodak, and a communications specialist with the U.S. Air Force.

Fletcher completed his studies at the Rochester Institute of Technology ■

Toronto

Jane Minett has been appointed general manager of SHARP APL timesharing in the United States. She brings to the position 12 years of experience with SHARP APL in systems design, consulting, marketing, publishing, and management. Her involvement with APL started at the University of Toronto where she worked in a graduate department providing support to faculty and administrative staff.

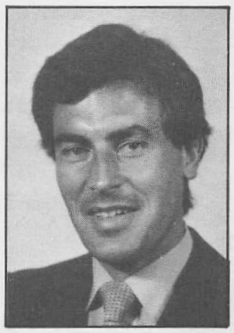
In 1973, Jane moved to Calgary, home of Canada's petroleum industry. She worked in the corporate planning department of a large oil and gas explorer and producer. Using APL, she computerized the company's project analyses carried out for its international operations.

In 1976, Jane joined I.P. Sharp in Calgary as branch manager. There she continued to spread the use of APL to pipeline companies, banks, transportation consultants, government, real estate developers, and, of course, to more energy companies.

In 1980 Jane moved to I.P. Sharp's international headquarters in Toronto. In her new position, she was responsible for all literature published by the company for worldwide distribution, and, more recently, for the direction of I.P. Sharp's financial planning software.

Jane was born in Philadelphia and later moved to Rochester. She is a graduate of the University of Toronto with a B.Sc. degree in mathematics ■

Zurich



Werner Huber is the new branch manager in Zurich. Before joining I.P. Sharp, he was with Rank Xerox (Switzerland) for 13 years. He has considerable expertise with APL in the fields of financial, personnel, and performance planning as Manager of Technical Services ■

Denver

Bill Paxson, formerly branch manager in Dallas (*I.P. Sharp Newsletter, May/June 1982*), has been appointed branch manager in Denver ■

Rochester

Barbara Siebert, formerly technical manager of the Rochester office (*I.P. Sharp Newsletter, January/February 1982*), has been appointed branch manager in Rochester ■

New APL Organization in Europe

APL is gaining in popularity with both inexperienced users and computer professionals in Europe. As a result, Euro APL was recently formed, representing over 1 300 European APL users. The organization is sponsored by the EEC. Euro APL seeks to promote the further growth of APL in a European context.

This year, some of the association's plans include: translating the draft ISO APL standard into French, Dutch, and German; having Professor Janko of Germany compile information about APL on micros; and having Theo Formanek of Switzerland compile information about APL and graphics.

Euro APL has already arranged for the exchange of information among national APL user groups, and will establish formal links with SIGAPL, the corresponding American association.

Philip Goacher, chairman of the UK APL user group, will coordinate the activities of European implementors and arrange for their representation in the United States.

The board of Euro APL is chaired by Gilles Martin of France, and the secretary is Joseph de Kerf of Belgium. Press and publicity are handled by Romilly Cocking of the United Kingdom ■

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.

Name: _____

Title: _____

Company: _____

Address: _____

_____

_____

☐ Please send me information about your courses in:

City: _____ Tel.: _____

The *I.P. Sharp Newsletter* is published by I.P. Sharp Associates, Suite 1900, 2 First Canadian Place, Toronto, Canada M5X 1E3.
Your comments and contributions are welcome.

Editor: Irene Shimoda
Circulation: Mary Kopfensteiner
Printed in Canada
January 1983
ISSN 0226 854X

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

Brisbane
I.P. Sharp Associates Pty. Ltd.
11th Floor, 127 Creek Street
Brisbane, Queensland 4000
Australia
(07) 229 8330

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 Avenue S.W.
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 1
P.O. Box 452
SF-00101 Helsinki 10, Finland
(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) 730-4567
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 SA
(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 du Cirque
75008 Paris
France
(1) 225 98 20

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

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-Gu (CPO Box 2810)
8269 Seoul, Korea.
771-91/2
Telex: 23341-5 / 24295 DAEWOO K

Singapore (Far East H.Q.)
I.P. Sharp Associates(S) Pte. Ltd.
Suite 1601, CPF Building
79 Robinson Road
Singapore 0106
Republic of Singapore
Telex: 20597 IPSAS RS

Singapore
Singapore International
Software Services Pte. Ltd.
Suite 1601, CPF Building
79 Robinson Road
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
Telex: 2322008 INTECA J

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

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 Lane
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 303
155 Willowbrook Blvd.
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
CH-8001 Zurich, Switzerland
(01) 211 84 24

I.P. Sharp 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 • Austin • 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 Amsterdam and Rochester, New York nodes.