the I.P.Sharp newsletter

SEPT./OCT. 1978
Vol. 6 Number 5

TELEX ACCESS TO SHARP APL

David Chivers, London

TELEX TELEX TELEX TELEX

```
0057564 1442 070978
INTLX MTL CA +
I P SHARP TOR

PTS
4418859+
21.44
18859 ITS NL+
I P SHARP TOR
```

SHARP APL CAN NOW BE ACCESSED FROM THE WORLDWIDE PUBLIC TELEX NETWORK, USING THE STANDARD TELEPRINTER. NO SPECIAL MODEMS, ACOUSTIC COUPLERS, OR OTHER ATTACHMENTS ARE REQUIRED.

THE NORMAL USE OF TELEX IS TO TRANSFER WRITTEN COMMUNICATIONS FROM ONE TELEPRINTER TO ANOTHER OVER A SWITCHED TELEGRAPH NETWORK. MOST CALLS, INCLUDING INTERNATIONAL ONES, ARE SET UP BY AUTOMATIC SWITCHING EQUIPMENT, THE CALLER EITHER TYPING OR DIALING THE DISTANT SUBSCRIBER/S NUMBER.

ACCESS TO SHARP APL IS PROVIDED BY DIALING THE TELEX NUMBER OF A MINICOMPUTER. THE MINICOMPUTER ACTS AS A PRE-PROCESSOR FOR ESTABLISHING AND MAINTAINING A CONNECTION TO THE SHARP APL PACKET-SWITCHING NETWORK, AND HENCE TO ONE OF THE MAINFRAMES ATTACHED TO THE NETWORK. ONCE THE MINICOMPUTER HAS RECEIVED AND ANSWERED THE USER/S CALL, IT WILL RESPOND WITH AN ANSWERBACK CODE WHICH VERIFIES THAT THE CALL IS ESTABLISHED. THE USER THEN PROCEEDS WITH THE SIGN-ON SEQUENCE AND SUBSEQUENT DIALOGUE AS IF USING AN APL TERMINAL.

(continued on p. 2)

IN THIS ISSUE

TELEX (continued)

Before everyone throws out their APL terminals to switch to Telex, some of the disadvantages should be pointed out:

- the terminal speed is slow (6.7 characters per second).
- the character set is very limited: as little as 47 printable symbols (*A* to *Z*, 0 to 9, and only 11 others), which doesn't allow too many APL operators.
- the page width is limited to 69 characters.

These limitations will restrict the usefulness of Telex considerably, and it is certainly not recommended for those who wish to write programs in APL. Provided that the terminal use is normally confined to data entry for existing APL applications and a limited amount of report printing, Telex will be useful, particularly to:

- the multinational corporation wishing to expand APL access to all its branch offices, including those in countries the Sharp Communications Network has not yet reached. In some cases a Telex connection may be more economical for branch offices not within local call range of the nearest Sharp network access point, even though normal access is available within the same country.
- users with only small APL requirements, and who already possess a Telex machine. The user is saved the cost of acquiring a special APL terminal.

Since the teleprinters are normally equipped with paper tape readers and punches, Telex access will provide a simple means of transferring APL output to punched paper tape, and for entering to APL paper tapes prepared either off-line on the teleprinter or by other computer systems. Off-line preparation and verification of paper tapes will reduce the connect time required for data entry applications. To allow paper tapes to be read, we have implemented a special modification. A user connected to APL normally types each line of input, followed by a carriage return, then waits for APL to respond with a linefeed or bell - the signal to proceed with the next line. The paper tape reader cannot be stopped at the end of line, but runs continuously once started. To overcome the problem, input from Telex is buffered in the interface minicomputer, and bells and linefeeds can be suppressed.

To allow the essential APL operations, some of the characters of the standard Telex character set are treated differently from the normal Telex keyboard characters. All APL Telex users will have keys for . , (← → + ¯ ∨ ∧ ? and (simulated) backspace, and many users will also have $ ⎕ and +.

Access via Telex has been provided at the Amsterdam office (Intersystems B.V.) - the access number and answer-back is 18859 ITS NL.

For those who, despite the warnings given above, must try to write programs or execute other APL operators at a Telex terminal, a useful workspace is available. Workspace 5 *FONT* provides function definitions which are equivalent to the APL operators for which no symbols exist on the Telex keyboard. The workspace also provides a means of editing functions by translating the function text into a representable form (using character strings beginning with ? for all non-printable characters) which can then be edited with an editor similar to 4 *EDIT*, before being translated back into a function definition.

Further details can be obtained from your local I.P. Sharp representative, or from Asoka Nimalasuriya in Amsterdam, or Simon Garland and David Chivers in London.

SARTEMS - SHARP APL RELATIONAL TEXT MANAGEMENT SYSTEM

SARTEMS is a text management system which provides facilities for the **storage**, **correction** and **retrieval** of non-numeric data. The system is, however, equally able to handle numeric information. It is comprised of a number of conversational APL functions which enable the user to establish a data base appropriate to a particular application, and to manipulate and query the data. Using SARTEMS, those with little or no knowledge of APL are able to make use of the powerful data-handling capabilities of SHARP APL.

It is an excellent system for organizing and storing related data such as directories, glossaries, indices, or bulky bibliographies that will be referenced frequently, since the retrieval costs do not go up as the amount of data increases.

Data is stored in entities known as **relations**. A relation is a mathematically based structure which can be conceptualized as a table or matrix, with a number of rows and columns. Each row in a relation is called a **tuple** and corresponds roughly to the traditional magnetic tape record. For example, a tuple in a telephone directory would consist of a person's name, address and telephone number. Each column in the relation is known as a **domain** and corresponds roughly to the concept of a field within a record. The first domain in a telephone directory would consist of all names in the directory. Each tuple must account for the total number of domains in the relation.

One important SARTEMS feature enables you to set up a "network" of data using **extended relations**. An extended relation is one which points to one or more secondary relations, each containing information relevant to the data in the first. This means that, when retrieving data, you can access the information contained in several relations at one time.

In traditional data files, fields tend to have a fixed, pre-defined length. SARTEMS allows domains to be of variable size, or to have no value (really the null value). The concept of variable domain width is consistent with the manner in which the SHARP APL File System handles data. In fact, each relation is a file and each tuple is stored as a separate component within that file.

There are two distinct phases in the development of a SARTEMS data base: data base design and data entry/manipulation. In the first phase, the data base designer establishes the relations appropriate to a specific application. This involves building a skeleton of named relations and specifying the characteristics for each domain in the relation (i.e., domain name, data type, and so on). Certain constraints can also be imposed on the data for checking or protection. For example, the data base designer can stipulate that the data to be entered for a particular domain must exist in a pre-determined list of words, or within a pre-defined numeric range.

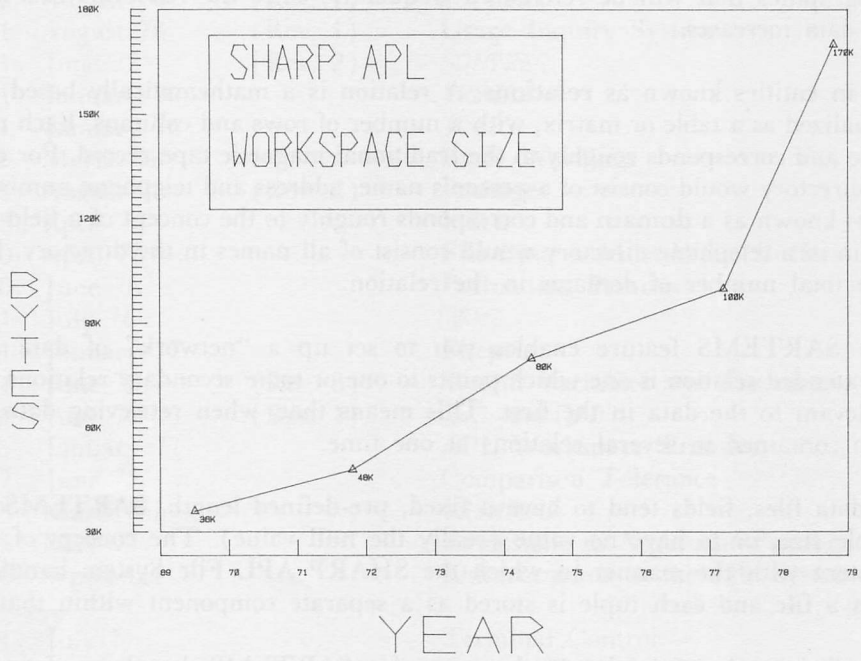
In the second phase, data is entered for each relation which has been established in the first phase. This data can then be retrieved, modified and displayed. Seven easy-to-use report writing functions are available for producing reports of the data (or a subset of the data) contained in a relation. These reports can be printed either at your terminal or on the highspeed printer. In addition, the *QUERY* function lets you perform sophisticated queries on a relational data base. Any specified subset of data can be accessed and displayed, or brought directly into a "work file" in order to produce reports using your own APL functions.

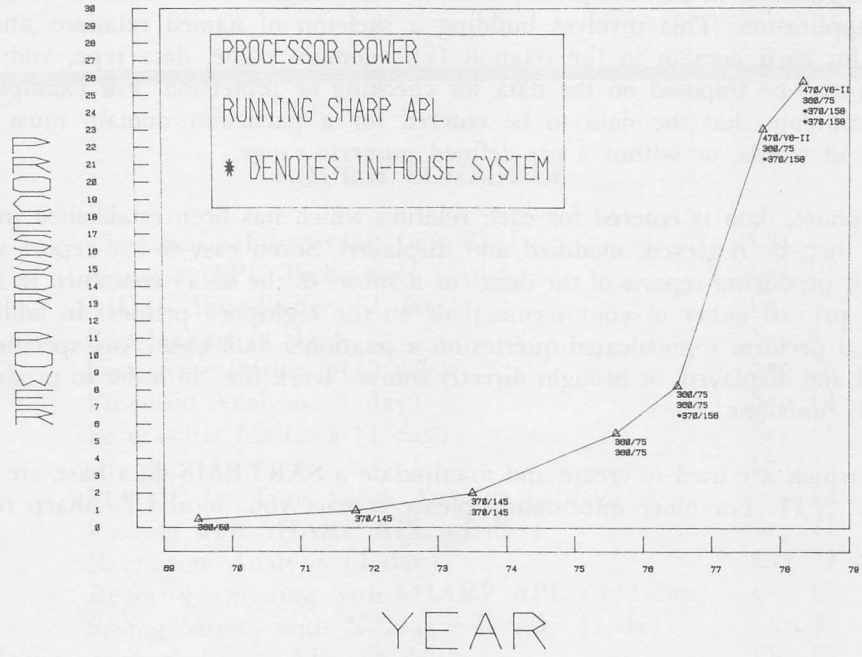
The functions which are used to create and manipulate a SARTEMS data base are contained in the workspace `587 TEXT`. For more information please contact your local I.P. Sharp representative.

WORKSPACE SIZE INCREASED TO 170K

Eric Iverson, Toronto

Recently we upgraded both the AMDAHL V6-II and the IBM 360/75 to be 3 megabyte machines. This allowed us, on the fifth of August, to increase the workspace size from 100K to 170K bytes. The graph "SHARP APL WORKSPACE SIZE" shows the history of the growth of workspace size (and provides some tantalizing thoughts for the future). The second graph, "PROCESSOR POWER RUNNING SHARP APL", documents a different, but equally interesting, dimension of growth.





SECURITY: RISK ANALYSIS

David Bonyun and Alistair McKinnell, Ottawa

Although computer security is of general concern, a strict, systematic approach to the problem and its solutions is rare. A formal protocol was formulated at a course given earlier this year in Ottawa to the Department of National Defence. Since then, risk analysis has been further refined and will be presented in the Phase Two Bracketting report due to be completed at the end of September 1978. The systematic approach to risk analysis is briefly outlined below.

The primary topics, or concepts, of risk analysis are **assets, threats, exposure,** and **protective mechanisms**. Secondary topics are used to calculate and evaluate the primary topics and these topics consist of **threat frequency, event cost, realization cost** and the **cost-benefit evaluation** of protective mechanisms.

Assets: Assets consist of any items of value belonging to an organization. Any asset belongs to one of ten gross asset categories: Hardware, Service, Data, Software, Personnel, Test Data, Communications, Physical Space, and Documentation. Each asset has certain costs and values associated with it so that its value can be determined when it is compromised. The scope of this brief does not allow for more detail here.

Threats: Threats are any occurrences which can compromise an asset. Threats belong to one of five gross threat categories: Disclosures, Theft, Contamination, Interruption and Destruction. Threats of a certain gross threat category can only effect certain asset types. This relationship is shown in the chart below.

Categories	Disclosure	Theft	Contamination	Interruption	Destruction
Hardware		×	×		×
Service		×		×	
Data	⊛	⊛	×		×
Software	⊛	×	×		×
Personnel				×	
Supplies		×		×	×
Test data	⊛	⊛	×		×
Communication				×	
Physical space					×
Documentation	⊛	⊛	×		×

× - threat exists in this area

⊛ - threat exists but event does not necessarily cause the asset to be lost.

Exposure: Exposure is a concept used to determine the severity of threats and to evaluate the effectiveness of protective mechanisms. Exposure is the expected loss, in dollars per year, due to a threat. It is used before the introduction of protective mechanisms to scale threats according to severity. Following the introduction of protective mechanisms the change in exposure indicates the effectiveness of the introduced mechanisms. Exposure is the product of the threat frequency and the event cost (two secondary topics).

Protective Mechanisms: Protective mechanisms may be implemented by any policy software, hardware, or procedures where mechanisms reduce exposure caused by threats. Reduction in exposure is accomplished by lowering either the threat frequency or the event cost.

Threat Frequency: Threat frequency is the expected number of occurrences of a threat per year.

Event Cost: The event cost is the cost to an organization when a threat takes place and compromises one or more assets.

Realization Cost: The realization cost is the cost an agent incurs when bringing about the occurrence of a threat. An increase in the realization cost causes a lowering of the threat frequency and hence the exposure.

Cost-Benefit Analysis: Cost-benefit analysis is used to evaluate the effectiveness of protective mechanisms. The costs consist of the initial and maintenance costs of the protective mechanism. Benefits consist of the resale value of the mechanism (if any) and the reduction in exposure. The tools of cost-benefit analysis allow for accurate comparison of mechanism effectiveness.

Order of Evaluation: We suggest the usage of the following sequence in risk analysis evaluation: delineate all assets of the organization in question, determine as precisely as possible all threats to the delineated assets, calculate a current exposure using the current threat frequency and event cost, propose protective mechanisms to cover threats and reduce exposure, calculate the new exposure using the new threat frequency and event cost, and ended by realization cost calculations. Lastly, determine the effectiveness of the proposed protective mechanisms using a cost-benefit analysis.

We believe that this is the first systematic attempt to obtain results in this area by combining the two perspectives of risk analysis and cost benefit analysis.

THE BIGGEST APL DATA BASE IN THE WORLD

Dave Keith, Toronto

While this claim has not been verified by the Guiness Book of world records, it must surely be true that, in terms of sheer size for a single APL data file, the ER586 data base on the SHARP APL system ranks up at the top. ER586 is a data base containing detailed information on all flights flown in scheduled service by the U.S. airlines, and contains figures right down to the level of flight number and type of aircraft. The actual data file contains over 300,000 components of data - each component containing 36 monthly time series, starting in January 1973. This translates to over 10 million time series in the same file. Assuming that each number in the data base is an integer (4 bytes), the file size works out to be 300,000×36×72×4 or 3,110,400,000 bytes. In actual fact, using a data compaction technique described below, the real file size is slightly over 500 million bytes. Associated with the data file is a directory file of 10 million bytes, so that instant access to any of the data in the data file is possible with a minimum of file reads.

There are several items related to the creation, updating and maintenance of ER586 that should be of general interest. The first of these is that, if you plan a similar venture, you had better first make friends with your local I.P. Sharp Associates branch manager -- file storage costs might be prohibitive. As an on-line SHARP APL file it has the advantage that any part of the data is instantly available (via MAGIC in public library `702 ER586`), but has the disadvantage of being much more expensive than storing its equivalent on tape in a storage room (approximately 30 tape reels).

In many ways, the SHARP APL File System is a perfect device for storing this data. Each component in the data file contains data for every unique origin, destination, carrier, equipment-type, and flight number. Since January 1973, there have been just over 300,000 of these unique occurrences, as mentioned above. The components are effectively 36 by 72 APL matrices -- there are 36 accounts, or rows of data, by 72 monthly time periods. But in actual practice many airlines juggle flight numbers and equipment from month to month, with the net result that a high percentage of these matrices are quite sparse. Using the SHARP APL data type known as packages, it becomes quite easy to store only the non-zero parts of each matrix. For example, if *M* is a 36 by 72 APL matrix, and it is desired to place the non-zero portion of *M* into component 5000 of the file with file number 2, then the following suffices:

```
B←M≠0
V←(,B)/,M
(⎕PACK 'B V') ⎕REPLACE 2,5000
```

The binary matrix *B* is created to record the positions of the non-zero elements in *M* (a 1 marks a non-zero position, and a 0 marks a zero position). Since binary APL variables occupy one bit per element, the physical size of *B* is 36×72=2592 bits, or 324 bytes. Next, a vector *V* is established which contains the actual non-zero elements of *M*. Finally, a package containing only *B* and *V* (but not *M*) is placed in the data file. If *M* originally contained only 10% non-zero data, then it would occupy 10,368 bytes as an APL variable in the workspace, but only 1360 bytes in the file as a package. In order to actually use the data if this package is referenced by the APL access function, it must be converted back to a rectangular 36 by 72 matrix, which is accomplished via:

```
⎕PDEF ⎕READ 2,5000
M←(ρB)ρ(,B)\V
```

Updates to the ER586 data base are performed monthly using a tape provided by the Civil Aeronautics Board. An update for a single month generally requires over three hours of elapsed time to complete. The first phase is a COBOL program which primarily prepares an APL feed file. The second phase is an APL-based function which performs the update. On an average month, data for 20,000 flights is posted into the file. Some of the data is for flights that have already occurred for previous months (in which case ⎕*READ* and ⎕*REPLACE* activity is required), and some for new flights (for which new 36 by 72 matrices must be established and put into the file using ⎕*APPEND*). Due to the nature of the SHARP APL File System, the ⎕*REPLACE* can cause a temporary problem. Each file component that grows by even a single number requires an extra 6K of file space. Thus, although the feed file may occupy only 8 million bytes; it temporarily causes the data file to grow by approximately 100 million bytes. Each Friday night, all files on the SHARP APL system go through a full restore, so that the net effect of an update on the data file is a growth of about 8 million bytes, as expected.

Several problems relating to the efficiency of data access and cross-referencing have been successfully solved with respect to this data base. The directory mechanism chosen is one where the origin and destination airports are the primary keys. Thus, with a maximum of three file reads, it is possible to return all the information on a flight such as United Airlines DC-10, flight number 123, from Chicago to San Francisco, for all periods since 1973.

In fact, this would be a very cheap operation using APL and a very expensive one for a batch system, since the batch system would have to scan 30 tape reels to come up with the same information. Interestingly, the file design does not necessarily adversely affect those interested in data for a single month only, since the cost of a ⎕*READ* does not change much with the amount of data read.

The following output, showing detail by flight number for all flights on scheduled service between Los Angeles (LAX) and Seattle (SEA) for the month of February, 1978, is a good example of what can be done with the ER586 data base. The first section contains a summary of the two-way traffic for each of the carriers involved -- Western (WA), United (UA), Flying Tiger (FT) and Hughes Airwest (RW). The remaining two sections show directional detail by flight number. Flight RW725 illustrates the fact that ER586 is an "as flown" data base. RW725 is normally scheduled to fly from Spokane to Los Angeles, but for one day in February was rerouted via Seattle.

It is only natural that the Official Airline Guide data base (see the next article) will be an excellent companion data base to ER586.

FEBRUARY, 1978

				TRANSPORTED				ENPLANED AT ORIGIN			CARGO TONS TRANSPORTED		
	CARRIER/	DEPARTURES		PASSENGERS		LOAD-FACTOR		PASS.	PASS.	°/°		FREIGHT	WEIGHT
SEGMENT	FLT. NO.	SCH.	PERF	COACH	TOTAL	COACH	TOTAL	ENPL.	DEPL.	LOCAL	MAIL	+EXPR.	L.F.
LAX-SEA	WA	196	195	17,091	18,349	53.7	49.0	16,196	11,636	71.8	92.5	409.0	45.1
	UA	387	382	24,934	27,011	55.2	53.0	26,344	26,344	100.0	72.6	239.3	44.0
	FT	0	4	0	0	0.0	0.0	0	0	0.0	0.0	64.7	18.1
	RW	0	1	63	63	61.2	61.2	0	0	0.0	0.0	0.0	48.5
	TOTAL:	583	582	42,088	45,423	54.6	51.3	42,540	37,980	89.3	165.2	713.0	43.7
LAX→SEA	WA 622	28	28	2,754	2,969	81.3	79.7	1,849	1,849	100.0	1.5	26.8	67.4
	WA 719	28	26	1,652	1,777	33.8	30.2	1,777	1,124	63.3	52.8	105.8	40.7
	WA 731	24	24	2,510	2,706	50.3	44.4	2,439	2,439	100.0	7.7	145.7	49.0
	WA 735	4	4	339	367	40.7	36.1	291	291	100.0	0.5	14.2	35.6
	SUBTOT WA	84	82	7,255	7,819	51.5	46.8	6,356	5,703	89.7	62.4	292.5	49.1
	UA 290	28	28	1,575	1,694	64.1	62.1	1,694	1,694	100.0	6.9	2.9	50.8
	UA 292	27	27	1,361	1,438	58.6	55.5	1,149	1,149	100.0	5.4	7.5	46.5
	UA 294	28	28	1,886	2,065	77.3	75.9	2,065	2,065	100.0	5.5	3.7	0.9
	UA 308	28	26	1,707	1,822	73.2	70.3	1,822	1,822	100.0	6.7	2.7	6.5
	UA 402	27	27	1,258	1,484	21.4	22.1	1,484	1,484	100.0	22.3	85.8	27.5
	UA 781	27	27	1,757	1,862	75.7	71.8	1,828	1,828	100.0	1.6	9.3	58.5
	UA 855	28	28	3,328	3,645	76.6	72.7	3,301	3,301	100.0	0.1	7.1	6.0
	SUBTOT UA	193	191	12,872	14,010	58.3	56.1	13,343	13,343	100.0	48.4	118.9	47.3
	DIR.TOTAL	277	273	20,127	21,829	55.6	52.4	19,699	19,046	96.7	110.9	411.4	48.0
SEA→LAX	WA 603	28	28	2,322	2,466	68.6	66.2	2,466	0	0.0	0.1	13.8	53.9
	WA 615	0	1	65	65	62.5	62.5	65	34	52.3	0.3	0.2	63.0
	WA 621	28	28	2,884	3,042	85.1	81.7	3,042	1,901	62.5	24.2	16.4	71.5
	WA 631	28	28	2,560	2,750	44.0	38.7	2,750	2,750	100.0	0.1	42.1	31.4
	WA 722	28	28	2,005	2,207	39.8	36.2	1,517	1,248	82.3	5.3	44.1	30.8
	SUBTOT WA	112	113	9,836	10,530	55.4	50.7	9,840	5,933	60.3	30.1	116.5	41.9
	UA 337	27	25	2,653	2,854	50.9	46.5	2,854	2,854	100.0	1.6	33.8	37.2
	UA 347	28	28	1,832	1,957	56.5	55.6	1,957	1,957	100.0	5.5	10.4	44.2
	UA 353	28	28	1,730	1,829	70.4	67.1	1,829	1,829	100.0	0.4	5.6	53.5
	UA 373	28	28	1,912	2,043	59.0	58.0	2,043	2,043	100.0	1.9	4.6	44.1
	UA 381	27	27	1,301	1,385	56.0	53.4	1,385	1,385	100.0	1.5	19.0	47.2
	UA 383	28	26	1,584	1,707	69.8	67.5	1,707	1,707	100.0	1.8	3.5	53.5
	UA 407	28	28	1,050	1,226	24.2	24.5	1,226	1,226	100.0	11.5	35.0	25.5
	UA 4909	0	1	0	0	0.0	0.0	0	0	0.0	0.0	8.5	22.3
	SUBTOT UA	194	191	12,062	13,001	52.2	49.9	13,001	13,001	100.0	24.2	120.3	40.8
	FT 641	0	1	0	0	0.0	0.0	0	0	0.0	0.0	21.7	43.5
	FT 3070	0	2	0	0	0.0	0.0	0	0	0.0	0.0	24.5	11.9
	FT 3072	0	1	0	0	0.0	0.0	0	0	0.0	0.0	18.5	18.0
	SUBTOT FT	0	4	0	0	0.0	0.0	0	0	0.0	0.0	64.7	18.1
	RW 725	0	1	63	63	61.2	61.2	0	0	0.0	0.0	0.0	48.5
	DIR.TOTAL	306	309	21,961	23,594	53.6	50.3	22,841	18,934	82.9	54.3	301.5	40.1

ACCESS TO THE OFFICIAL AIRLINE GUIDE DATA BASE

Bob Dabrowski, Toronto

In last month's issue there was a short discussion of the OAG data base. The examples below give an indication of the range of available facts, and of the ease with which the access functions extract and present the information. The data base has not yet been released.

There are three access functions to the OAG data base. They are *OAG*, *XOAG* and *ΔOAG*. *OAG* is used to extract time series results of a known size and is meant to be used in conjunction with the ER586 data base. *XOAG* serves as the cross reference function and within bounds can answer most any question you might have on scheduled airline traffic. *ΔOAG* is used to translate codes into their corresponding names. With the exception of *ΔOAG* all access functions depend on the time settings in MAGIC. The OAG data base can only be accessed through MAGIC.

The data base contains only September '78 data at the moment but will be expanded rapidly as we start up. The data for each month has both international and North American flight information. For each flight (some 200,000 each month) 21 items of information are retained. They include, among others, routes flown, service provided, flight times and equipment used (see the list below). The data base is current with the OAG issues as published by Reuben H. Donnelly Corporation, and each month's becomes available 9 days prior to the 1st of the month. There is a royalty fee to be paid to Donnelly for use of the data base and a contract must be signed prior to commencement of usage.

The access function *OAG*

OAG returns a **time series** result in a manner similar to the ER586 (service segment data base) access function. The syntax is:

R←**origin-destination** *OAG* **carrier,account,equipment,flightnumber**

For example, to display the number of departures and the departure time each month for flights 68, 60 and 62 (CP AIR) between Toronto (YYZ) and Montreal (YUL), type:

```
'YYZ→YUL' OAG CP,7 8,F 68 60 62
```

Some parameters are optional and if left out have special meaning. Only the origin-destination and accounts must be provided. When a parameter is omitted, multiple flights occur as a result. These flights are summarized so that only one result is returned. Compression also occurs if more than one flight fulfills the conditions of the access. In the CP AIR example there are really two flights with the number 68: one flies at a different time on Sunday. The information on the two flights was merged into one line on the basis of the one that flew most often. The departures per month were added and the weekday depart time chosen. There are summarization rules for each account and unless they are changed during the access the defaults will be used.

To find out whether compression has occurred, look at the variable *OAGCOUNT*. It has the same size as the result returned from *OAG* and contains the count of the flights that went into making up your answer. Compression is more obvious when you make an access like this one:

```
'YYZ→LHR' OAG 7,EQP '∊747'
```

Here we want to know the total number of departures for any type of 747 (*EQP* '∊747') from Toronto to London Heathrow (LHR). All flights regardless of carrier and flight number would be compressed. The count would be available in *OAGCOUNT*. Note that *OAG* uses a function named *EQP* to designate equipment types. This differs from ER586 which uses the function *T*. In this case *EQP* '∊747' means the family of all 747's.

The access function *XOAG*

XOAG is used to cross reference the OAG data base. It operates in a manner totally different from *OAG*. The result of *XOAG* is a **character matrix** and is obviously not in time series form. Where *OAG* allows you to specify only a limited number of parameters *XOAG* permits you to address the whole data base and all 21 accounts. The syntax for *XOAG* is:

R←**accounts** *XOAG* **specification statements**

For example, if you wanted to list all the cities to which you can fly from Toronto, you would enter:

```
CITIES←2 XOAG 1 EQL 'YYZ'
```

The number 2 to the left refers to account 2 - the destination city. The phrase to the right - 1 *EQL* *'YYZ'* - means you wish to consider only those cities which have an origin city (account 1) equal to YYZ. The result would be a character matrix of all the unique city codes that qualify.

Specification statements can be strung together. For example:

```
CARRIERS←3 XOAG (1 EQL'YYZ'),2 EQL'LHR'
```

answers the question "who flies between Toronto and Heathrow?". When more than one phrase appears an implicit AND is assumed between them. If more than one parameter appears within a phrase an implicit OR is assumed between parameters. So that (3 *EQL* *'CP,AC,ND'*), 5 *EQL* *'72S'* can be read as carrier is (CP **or** AC **or** ND) **and** equipment is 72s.

XOAG permits you to partition the database on the basis of the data itself. All accounts are permitted in specification phrases. If you wanted to know which flights flew to Rome (FCO) from Heathrow after 10 am and before 2 pm you would ask:

```
FLIGHTS←3 4 5 8 XOAG (1 2 EQL 'LHR∘FCO'),(8 GTE 1000), 8 LTE 1400
AE∘ 784∘DC8∘1615
AI∘ 102∘747∘1500
AI∘ 104∘747∘1330
BA∘ 850∘747∘1420 (etc.)
```

The result is a character matrix with a ∘ (upper case *J*) separating the columns. The accounts are returned in the order that they were asked for and all duplicate rows are removed.

The comparison operations allowed are *EQL*, *LTE*, *LTN*, *GTE*, *GTN* and *ELM*. The last, *ELM*, is an element operator. Some accounts are really composites in that they consist of a vector packed into one element. A good example is the days of operation of a flight. *ELM* allows reduction of the result on the basis of the elements within a field. To determine who flies through Denver (DEN) on their way to Los Angeles (LAX) on Mondays and Thursdays ask:

```
WHO←3 XOAG (2 EQL'LAX'),(18 ELM 'DEN'),6 ELM 1 4
```

Account 18 is the complete route of a flight.

The access function *ΔOAG*

ΔOAG is a mechanism to translate the various mnemonics used in the OAG data base. Some accounts such as cities, equipment, cargo service and even days of the week use codes. Supplying *ΔOAG* with the account number and the code will result in a translation.

```
NAME←2 ΔOAG 'YYZ,LHR'
```

will extract the names of the two airports.

Technical Supplement – 17

FUNCTIONS ADLIB

Stephen Taylor, Copenhagen

The transition from writing APL programs to writing APL systems is not always an easy one. A terminal interface subsystem, 999 *TASK* (see below and Jan/Feb Newsletter), was written for programmers who can't draw on a long history of programming encoded into a *UTIL* workspace. In the same would-be-helpful vein, 999 *ADLIB* is now offered. It is compatible with TASK and is intended to complement it. We have several applications successfully using both TASK and ADLIB in Copenhagen.

ADLIB is a clean, simple and powerful system for keeping APL functions on file. It is intended for programmers without extensive APL systems experience who are faced with writing a larger system than will operate comfortably in a single workspace.

The simple solution to this problem, and the one adopted by ADLIB, is to divide the system programs into three groups:

1) **A controlling structure** which initialises the system when the workspace is loaded and which controls what work is to be done.

2) **System modules**: groups of functions which do the main work of the system, and are called by the controlling structure (1).

3) **Utility functions** used by both (1) and (2). These may be specific to the system, or of general application.

ADLIB provides a Libraryfile for storing the system modules, functions for maintaining and documenting these modules, and functions for dispatching the modules within a system.

ADLIB can be useful even when all the code for the system will fit into one workspace. It offers several advantages:

- More room in which to execute functions. The code may fit into one workspace, but still not have enough space to run in. ADLIB helps you to avoid *WS FULL*s.

- Some SHARP APL operators use more efficient algorithms when more workspace is available. Increasing the amount of free workspace can reduce the cost of executing your code.

- Systems written using ADLIB are forced into a simple modular structure. This can be of particular benefit to novice systems-writers.

- ADLIB allows you to document your system modules individually, showing when they were last updated, and by whom. Documenting changes to your code is less trouble than with *WSDOC* or *EASYWSDOC*.

  "A hammer is an excellent tool, but nobody fishes with it." (Zen saying)

- ADLIB allows several programmers to work on a system simultaneously. A programmer can make corrections to a system module without having to *)SAVE* the workspace. Because of this, he does not need access to the workspace's account number.

- ADLIB keeps a copy of the "previous version" of each module, allowing fast and selective backup when an "improved" version goes wrong.

- ADLIB is an excellent starting point for cleaning up old APL systems.

- ADLIB allows you to use locked code for running your system without keeping a separate unlocked version for maintenance.

In practice, modules from an ADLIB Libraryfile are called and erased by the two functions *ΔPREAD* and *ΔPERASE*:

```
      ∇ ICALL
[1]   ⍝ calls and executes input module
[2]   ΔPREAD 16   ⍝ get module from libraryfile
[3]   IMAIN   ⍝ "top" function in module
[4]   ΔPERASE
      ∇
```

ADLIB includes a set of functions for maintaining and documenting the contents of the modules. A maintenance session to alter the function *IMAIN* called above might look like:

```
      GET 'INPUT'
      ∇IMAIN[4⎕12]
[4]         etc.
      ∇
      SAVE 'INPUT'
      PRINT 'INPUT'
```

(Output not shown.)

The ADLIB system can be found in 999 *ADLIB*. Full documentation is available both on-line and from the highspeed printer via functions in the workspace. The system is supported by its authors, Stephen Taylor and Johnny Larsen, who would be glad to answer any questions.

TASK FORCE

Stephen Taylor, Copenhagen

It's the systems designer's North West Passage, the dream that hangs around the edge of his consciousness. The Final System: the perfectly general system which only needs 20 parameters to be any system you could possibly want. Or would that be 100 parameters? 500?

If it ever gets written, it is sure to be in APL; though like the North West Passage, you probably wouldn't want it if you had it. (An icebreaker finally made it round to the Bering Straits, but reported the passage to be "of no commercial use".) However, a lot of useful things were turned up by the search for the Passage; Canada to name but one. This article is about something useful that turned up on the way to somewhere else.

⎕ is APL's "window on the world". For many systems, it's open too wide. In commercial applications, incoming data must be validated, and errors must be reported intelligibly. Workspace 999 *TASK* was offered (see Jan/Feb Newsletter) as a plug-in alternative to using ⍞ and your own checks. Instead of an open window there's a choice of bulletproof cashier's windows which only let in real money, count the cash to check that it adds up to the sum on the deposit slip, tells the customer if it doesn't, and returns the cash to him for another try. (Even my laundromat doesn't do that.)

In this, and in a second article, I shall illustrate some of the ways in which TASK has been used to compress and organise the coding of input programs. I am indebted to Knud Stig Andersen of Novo Industri A/S, who has been responsible for many of the ideas. (The Vikings reached Greenland first, right?)

The first example deals with a familiar problem in an invoicing system. Data for the invoices has first to be input, and later updated or edited to correct errors. Now when the data is entered for a new invoice, the user should be prompted for each field in turn; but when editing an existing invoice, he should get the choice of which field to amend. The simplified program below manages to combine both the input and the editing conversations into a single function.

```
    ∇   GETDATA X.ANS
[1]     →(F1,OLD)[X] ⍝  <X> IS 1 FOR INPUT AND 2 FOR EDITING
[2]    OLD:→END ONNULL ANS←FIELDNAMES ΔASKO 'FIELD: ' ⋄ →⍎'F',⍕ANS
[3]    F1:→(END,OLD)[X] IFNULL ANS← 1  6 ΔASKI 'INVOICE NO: ' ⋄ DATA[1]←ANS ⋄ →(F2,OLD)[X]
[4]    F2:→(F1,OLD)[X] IFNULL ANS←CUSTOMERS ΔASKNO 'CUSTOMER: ' ⋄ DATA[2]←ANS ⋄ →(F3,OLD)[X]
[5]    F3:→(F2,OLD)[X] IFNULL ANS←⎕TS[1] ΔASKD 'DATE: ' ⋄ DATA[3]←ANS ⋄ →(F4,OLD)[X]
[6]    F4:→(F3,OLD)[X] IFNULL ANS→PRODUCTS ΔASKNO 'PRODUCT: ' ⋄ DATA[4]←ANS ⋄ →(F5,OLD)[X]
[7]    F5:→(F4,OLD)[X] IFNULL ANS← 1  2 ΔASKR 'AMOUNT: ' ⋄ DATA[5]←ANS ⋄ →(END,OLD)[X]
[8]    END:
    ∇
```

For this application, the trap in *ΔASK* for 'END' was changed to trap the null response (just hitting carriage return) and the *IFEND* function retitled *IFNULL* to reflect this.

Entering the function with *X* set to 1 for input, we branch immediately to *F1*, the prompt for the first field. *ΔASKI* prints the prompt given as its right argument, and waits at the end of it for your reply. You can type what you please, but because of the left argument, everything will be rejected except a single 6-digit number, or a null response. *IFNULL* traps a null response, and from here it would send you to the end of the function. Otherwise, your 6-digit invoice number is stacked in *DATA*[1] and you are passed on to *F2* for the next field.

If you come in to edit however, (with *X*←2) line 2 would ask you which field you wanted to edit, and send you to it. Whether or not you gave a new value to this field, you would find yourself back at the 'FIELD: ' prompt on line 2 afterwards, and would keep coming back to it until you gave a null response to that.

This aspect of it is really quite pretty. During input, a null response drops you back at the previous prompt, so you can correct the invoice while you're putting it in, but you **have** to either finish it or back out entirely. During editing, the same lines of code keep sending you back to choose a new field.

Some of the functions in 999 *TASK* have been amended slightly or extended, and versions of some of these will be appearing in the workspace. A summary of the ones referred to here:

ANS←FIELDNAMES ΔASKO 'FIELD: '
: *FIELDNAMES* is a character matrix of options to be chosen, in this case the names of the fields in the invoice. *FIELDNAMES[ANS;]* is what the user chose.

ANS←CUSTOMERS ΔASKNO 'CUSTOMER: '
: *CUSTOMERS* is a numeric vector of customer codes. *ANS* is one of them.

ANS←⎕TS[1] ΔASKD 'DATE: '
: *ANS* is a date in 3↑⎕*TS* form, not before this year, and not later than the current date (by default).

ANS← 1 2 *ΔASKR 'AMOUNT: '*
: *ANS* is an integer vector of length 1. Dividing it by 100 (10*2) will give you the number typed in.

Apart from the fact that it's short, and therefore both quick to write and easy to maintain, *GETDATA* has the advantage that, since the same code is used for input and editing, you **know** the same validation is being performed for each. And you only have the one function to maintain, not two in parallel. Because you're using the TASK functions, you know you automatically have consistent error handling, and user support built in. You only have to supply the prompts and the 'help' messages.

In the next article I shall show how a single *ASK* can be constructed to prompt for and validate dozens of fields in a record. (Readers who are interested in TASK are invited to contact the author at the Copenhagen office.)

*** WARI CONTEST RESULTS ***

Jerry Cudeck, Toronto

As the sun sinks majestically in the west, and the tribesmen slowly make their way back to their villages, the sound of jungle drums proudly proclaim our own system librarian, **Ed Stubbs**, to be the greatest *WARI*or of them all! Ed battled courageously with combatants especially groomed for the occasion by Dinos Appla, Mike Powell, Ron Bradley, and Wayne Harrington - all members of the Sharp global village. In recognition of Ed's *WARI* prowess, he will be receiving a book prize of his choice in the near future.

Each of the ten competing programs were pitted against the others in a round-robin style tournament, with each competitor playing all others twice - once as player number one and once as player number two, so as to ensure no unfair advantage of first move. The complete results of the competition are summarized in the two following tables. The first table shows which player, (either 1 or 2) was victorious in any particular match. The second table shows the number of victories recorded by each program while competing as either the first or second player.

\ PLAYER 2 / PLAYER 1	WARIΔDINOS1	WARIΔDINOS2	WARIΔDINOS3	WARIΔDINOS4	WARIΔEDS1	WARIΔEDS2	WARIΔEDS3	WARIΔPOW1	WARIΔRBR	WARIΔWHA1
WARIΔDINOS1		1	2	2	1	1	2	2	2	2
WARIΔDINOS2	2		2	2	1	1	2	2	2	2
WARIΔDINOS3	1	2		1	1	1	2	2	2	2
WARIΔDINOS4	1	2	1		1	1	2	2	2	2
WARIΔEDS1	2	2	2	2		1	2	2	2	2
WARIΔEDS2	2	2	2	2	1		2	2	2	2
WARIΔEDS3	1	1	1	1	1	1		2	1	1
WARIΔPOW1	1	1	2	1	1	1	2		2	2
WARIΔRBR	1	1	1	1	1	1	1	2		1
WARIΔWHA1	1	1	1	1	1	1	2	2	1	

Table 1

	FIRST MOVE	SECOND MOVE	TOTAL
WARI∆DINOS1	3	3	6
WARI∆DINOS2	2	4	6
WARI∆DINOS3	4	5	9
WARI∆DINOS4	4	4	8
WARI∆EDS1	1	0	1
WARI∆EDS2	1	0	1
WARI∆EDS3	8	8	16
WARI∆POW1	5	9	14
WARI∆RBR	8	7	15
WARI∆WHA1	7	7	14

Table 2

Ed's program, as well as that of Ron Bradley who placed second in the tournament, may be viewed in workspace 999 *CONTEST*. Many thanks and congratulations to all those who took part. For all of those who considered entering, but did not, we invite you to take a *SPIN* with our next contest.

*** CONTEST NUMBER 7 - *SPINNING SHELLS* ***

By way of introduction to our new contest, the concept of "shells" of a square matrix is presented below. In this discussion, the following assumptions have been made:

(i) ⎕IO←1
(ii) 2=ρρM ◇ 1==/ρM ◇ 0≠1↑ρM
(iii) The degree of M is equivalent to 1↑ρM

Definition:

The first or innermost shell of non-empty, square matrix M is the central element of M is of odd degree, or the central square of M if M is of even degree. The (T+1)th shell of M is then defined as the square made up of those elements which are adjacent to, and on the immediate exterior of, the square perimeter of the Tth shell.

Example 1:

```
      M←3 3ρι9
      M
1 2 3
4 5 6
7 8 9
```

The first shell of M is its central element: namely, 5. The second shell of M is the vector 1 2 3 6 9 8 7 4. Note in this case that M has only two shells.

Example 2:

```
      M←4 4ρι16
      M
 1  2  3  4
 5  6  7  8
 9 10 11 12
13 14 15 16
```

The first shell of M is its central square; namely 6 7 11 10. The second shell of M is given by:
1 2 3 4 8 12 16 15 14 13 9 5

Note: If M is a non-empty, square matrix and if N←1↑ρM, then the number of shells contained in M is given by ⌈0.5×N.

By "spinning a shell" we mean a cyclic rotation of that shell in either a clockwise or counter-clockwise direction. The amount and direction of the spin is controlled by an integer "spin factor" where a positive factor induces a clockwise rotation and a negative factor induces a counter-clockwise one.

Example 3:

With *M* defined as in example 2, if a spin factor of 2 is applied to the first shell (i.e. innermost square) and a spin factor of ¯1 is applied to the second, the resulting matrix *R* would be given by:

```
      R
2   3   4   8
1  11  10  12
5   7   6  16
9  13  14  15
```

With the preceding examples serving as a guide, the specifications of our *SPINNING SHELLS* contest are as follows.

Required: A dyadic explicit function of the form:

R←SF SPIN MAT

where *MAT* is a non-empty, square matrix (either character or numeric) and *SF* is either an integer scalar or an integer vector of spin factors.

Note: (1) **If** 1=⍴,*SF*, then the single spin factor is to be applied to all the shells of *MAT*.

(2) If *N*=⍴,*SF* for any positive integer *N*>1, then *N* is determined as *N*←⌈0.5×1↑⍴*MAT* and the spin factor given by *SF*[*T*] is the one to be applied to the *T*th shell of *MAT*. Thus, to produce the result *R* of example 3, one would invoke *SPIN* with a left argument of 2 ¯1.

Note: Packaged entries including the submittor's name and address should be appended to file 999 *CONTEST* no later than December 31, 1978.

Not all accounts have to be translated so the only ones available through *ΔOAG* are cities (1 and 2), carriers (3), equipment (5), passenger service (12), cargo service (13) and meals (14). There are 5 additional accounts that are only accessible though *ΔOAG*. They are: account names (0), city latitude and longitude (30), minutes ahead or behind GMT for each city (31), equipment QSI ratings (32) and equipment configurations by carrier and equipment type in use (33).

```
AVAILABLE←0 ΔOAG ⍳21
```

will return to you the contents of the following table:

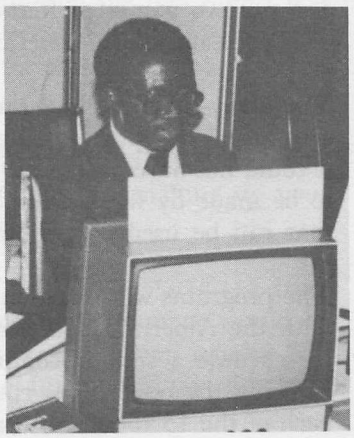
Accounts

1	Flight origin	12	Class of passenger service
2	Flight destination	13	Class of cargo service
3	Carrier code	14	Meals served
4	Flight number	15	Downline stops
5	Equipment code	16	QSI
6	Days of departure	17	Local route
7	Total number of departures scheduled	18	Through (beyond) route
8	Departure time	19	Exception/restriction note number
9	Arrival time	20	Effective (first) day
10	Total travel time	21	Discontinue (last) day
11	Total transit time		

INQUIRY SYSTEM AT THE CALGARY OIL SHOW

Frank Arthur, Edmonton

The bi-annual Canadian National Petroleum Show held in Calgary June 13 - June 15 was attended by close to 25,000 people, including visitors from countries such as Saudi Arabia, Venezuela, Mexico and the Soviet Union. To help match potential buyers with the appropriate companies the Alberta Department of Business Development utilized at the show an on-line data base system, developed by Roger Hui of the Edmonton office. With the help of a video display terminal, and an attached hard copy unit, the government officials were able to provide instantaneous sourcing information.



A show visitor interested in, say, oil and gas separators, would quickly be provided with a printout showing the Alberta companies that manufacture the product, together with their addresses, telephone and Telex numbers, contact persons and, if they were at the petroleum show, their booth numbers. Detailed information on any specific company in the data base could also be displayed. The system was especially helpful to international visitors who were not familiar with Alberta companies.

The data base currently contains information on some 260 companies in the oil and gas industry and there are plans to add companies in other industries. The petroleum show represented the official inauguration of the Computerized Inquiry System; the system has already proved itself to be a very useful tool and presents information quickly and effectively.

LEASE ANALYSIS

Linda Zetterstrand, Toronto

Over the last few years, equipment leasing has become an increasingly popular means of financing the use of productive assets. We are currently developing a package that will allow Canadian leasing companies to quickly evaluate and price prospective lease transactions.

The package will be comprehensive, handling simple and complex situations alike. Several alternative methods of dealing with tax implications will be present. Many different reports will be available, both for individual leases and for a portfolio of placed leases. Contact your local Sharp office for further details.

SHARP APL AT THE BRITISH COMMONWEALTH GAMES

Frank Arthur, Edmonton

The eleventh Commonwealth Games, held in Edmonton, Alberta, from August 3-12, saw over 1500 athletes from 47 countries participate in 10 different sports, including athletics, swimming and gymnastics. A sport that is rather unique to these games is lawn bowling, and the largest single international bowls competition ever held, took place at the Games. It is here that SHARP APL helped solve some of the problems faced by the Games organizers.

The three events in lawn bowling were singles, pairs and fours, with, respectively, 16, 14 and 15 participating teams. The activities took place on 4 bowling greens with 6 rinks to a green, and were held over a period of 8 days. One of the goals of the bowling organizers was complete fairness in the scheduling of the matches and the allocation of rinks. To this end a number of constraints were established:

- each team plays a maximum of 2 games per day.
- each team must stay on the same green for all its matches on a given day; however, no team should play on the same green for 2 consecutive days.
- no team must play more than one game on the same rink.
- no two games of fours should be played on adjacent rinks.
- all teams must have an equal share of games on the rinks at the edge of the green.

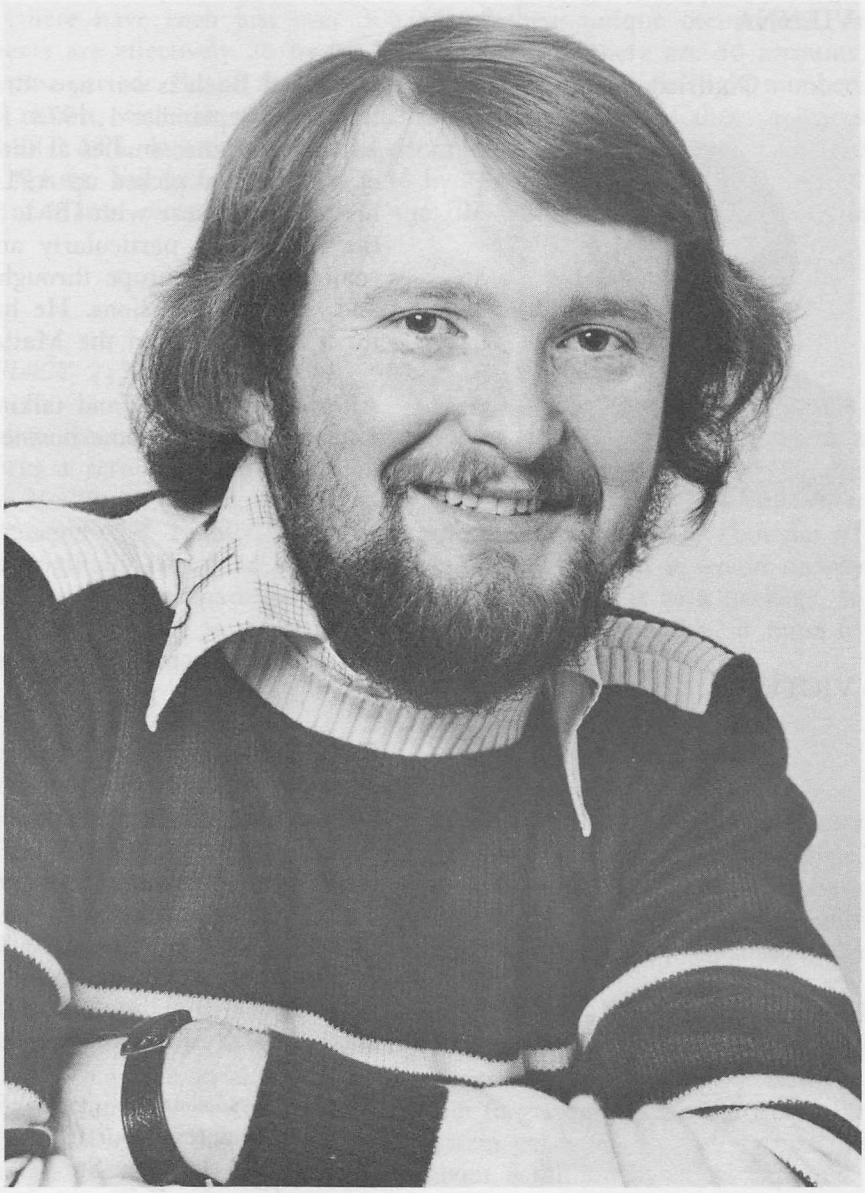
The array-handling and random-number generation features of APL made it a natural choice for the solution of this problem. Besides, the lead time for the development of the programs was so short that it could not have been done on time in any other programming environment. Since the draw was going to be made by the Games Bowls Technical Committee there was a need for a conversational system that can be used by people with no computer background.

The programs were written in a matter of days by Keith Toogood and Darwin Daviduk, programmers with the Alberta Department of Agriculture in Edmonton; the two had used APL earlier this year to schedule a similar competition in curling. Output from the programs included an event schedule showing the opposition, rink number and date of game for each team. Event and rink schedules for each of the 15 rounds of competition were also printed out. These printouts were the official schedules distributed to the team managers of all the countries.

SHARP NEWS

ABERDEEN

John Craig



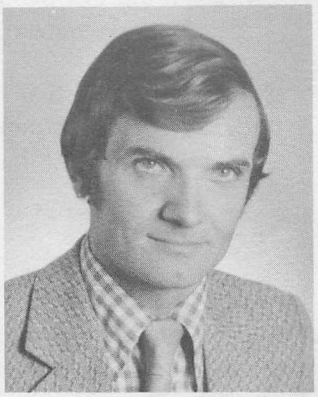
John Craig has just opened an office in Aberdeen, Scotland and invites you to call in if you are passing - "bring your woollies, it's cold up here". John joined the London, U.K. branch two years ago after completing a course in Maths and Computing at Paisley College of Technology, Scotland. He has since written several large multi-user management reporting and remote data entry systems for multi-national companies. His interest has been diverted to the use of APL in the simulation of hybrid and analogue computer systems.

The address of the Aberdeen office is:

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

VIENNA

Gottfried Bach

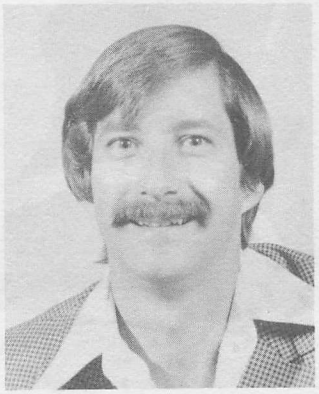


Gottfried Bach is our new Branch Manager in Vienna, Austria, as of September 1, 1978. He has been teaching mathematics and computer studies at the American International School of Vienna, and picked up APL four years ago while spending his sabbatical year with IBM. Since then, Gottfried has spread the APL virus particularly among the International Schools community in Europe through numerous courses, workshops and conference sessions. He has just finished the manuscript for a book "APL in the Mathematics Class".

After teaching APL and talking about it for so long, I think I am ready to do some business with it."

VICTORIA

Dave Smith



Dave Smith moved to Victoria, B.C. in July of this year to open our office there. Dave obtained his degree in Computer Science, and his first exposure to APL, at the University of Alberta, graduating in 1971. He worked for several years with the Alberta Provincial Government as a systems programmer doing teleprocessing system development. In 1977 he joined Sharp Special Systems as a senior consultant to install the Alberta PARIS System. With his recent move into APL support, Dave will maintain his interest in minicomputers, providing Special Systems support for the West Coast.

The address of the Victoria office is:

Chancery Courst,
1218 Langley St.
Victoria, B.C.
V8W 1W2
(604) 388-6365

AN AVIATION DATA BASE SEMINAR

On Nov 2, 1978, I.P. Sharp Associates is hosting an Aviation Data Base Seminar, being held at the Marriott hotel, Key Bridge, Washington, D.C. Our intention is to bring together for a day of informal discussion our current aviation data base clients with the rest of the aviation community. This seminar has been organized for executives and professionals concerned with all phases of the commercial aviation industry. It will cover a wide variety of subjects, including:

- The power of an on-line Official Airline Guide data base
- From problem to program: a consultant's use of APL and the Sharp Aviation data base
- Use of the Sharp Aviation Data Base in investment financial analysis
- The Sharp Aviation data base: history and future
- Can we see a future in the past? (Interpreting airline historic costs and performance)
- Market and route analysis
- Management reporting requirements for an intrastate airline

We invite you and other members of your organization to attend and participate. There is no registration fee. All attendees must be pre-registered. For more information regarding this seminar please contact your local Sharp office.

CAPERS

Frank Arthur, Edmonton

Workspace 57 *CAPERS* contains a set of programs for evaluating capital expenditure projects, using the discounted cash flow method. It can be used to analyze the economic feasibility of such things as plant expansions, real estate developments and other long-term capital projects.

For each project, the user defines a set of variables that represent the economic parameters, for example, the income for each year of the project, operating expenses, capital expenditures, depreciation information for tax purposes, and the applicable tax rate. Then, depending on the level of detail required, any one of three reports may be generated:

- a summary printout of profitability indicators (net present value, internal rate of return, etc.),
- a one-page cash flow report, and
- a 3-page detailed printout.

A data base system is also provided for storing input data on various projects and associated cases, for future evaluations or post-operational comparisons.

To obtain an 8-page documentation on how to use the system, type *CAPERSHOW*, or contact your local SHARP APL representative.

APPLICATION LIBRARIES UPDATE

NEW	57 *CAPERS*	See above.
CHANGED	1 *SATN*	See new index page 18.
	39 *X11*	The additive model is now fully operational. In addition, a new state setting *TTASK/NTASK/BTASK* has been incorporated with a view to reducing cpu costs. For further details, load the workspace and type *ΔDESCRIBE*.
MERGED	1 *PUBLICLIBS*	Now includes the contents of 1 *DEMOS* along with the other general library information.
	92 *GAMES*	Games workspaces 92 *GAMBLE*, 92 *SNIM*, 94 *WARGAMES*, 94 *WORDGAMES*, are now collected into a single workspace, 92 *GAMES*.
GONE	1 *DEMOS*	Contents in 1 *PUBLICLIBS*.
	20 *INDEX*	Obsolete documentation, (not saved in 499).
	92 *GAMBLE*	See 92 *GAMES*
	92 *SNIM*	See 92 *GAMES*
	94 *WARGAMES*	See 92 *GAMES*
	94 *WORDGAMES*	See 92 *GAMES*

NEW SYSTEM FUNCTIONS

New system functions, ⎕*FX*, ⎕*CR*, ⎕*EX*, ⎕*NC* and ⎕*NL* became available in June this year. Although these functions do not provide new capabilities (i.e. ⎕*FD* and ⎕*WS* can be used to model them), they are significant in that they exist in most APL systems (whereas ⎕*FD* and ⎕*WS* do not). Hence, they facilitate writing applications that can be moved from one installation to another. Moreover, they allow users of different systems to communicate more easily by extending the language that is common to all implementations.

The new system functions as described in SATN-20 (revision 3) are, briefly:

R← ⎕*FX M* ⍝ function establishment from a character matrix (similar to 3 ⎕*FD M*)

R← ⎕*CR N* ⍝ canonical representation (similar to 2 ⎕*FD N*)

R← ⎕*EX NM* ⍝ expunge (similar to 6 ⎕*FD NM*): *R* is a Boolean with 1's corresponding to valid names in the argument that are now available for use

R← ⎕*NC NM* ⍝ name classification with the following results:
- 0 - name is available
- 1 - label
- 2 - variable
- 3 - function
- 4 - other (groups, ⎕names, invalid names)

R←*A* ⎕*NL C* ⍝ name lists (similar to 1 ⎕*WS N*): *C* is a scalar or vector of classes as returned by ⎕*NC*; *A* is an optional scalar or vector designating the first characters of names to be returned

The System Functions described in SATN-14 (revision 2) are:

*P*1←*NL* ⎕*PEX P*2 ⍝ expunge objects named in *NL* from *P*2 (complementary to ⎕*PSEL*), and return *P*1.

R←*NL* ⎕*PNC P* ⍝ name classification within the package *P*. Results of ⎕*PNC*:
- ¯1 - object undefined but name in use
- 0 - object not in package (name available)
- 1 - (reserved)
- 2 - variable
- 3 - function
- 4 - invalid name

R←*NL* ⎕*PVAL P*⍝ returns the value of the variable named by *NL* within the package *P*.

SATN-INDEX

SATN-0	01	January 76		SATN Introduction
SATN-1	01	January 76		TASKID
SATN-2	15	September 77	(Rev. 2)	Control Messages
SATN-3	01	January 76		*⎕OUT*
SATN-4	01	April 78	(Rev. 2)	N-tasks and B-tasks
SATN-5	01	August 78	(Rev. 2)	Batch APL
SATN-6	01	January 76		Execute
SATN-7	01	January 76		Latent Expression
SATN-8	15	August 76	(Rev. 1)	*HSPRINT*
SATN-9	01	August 78	(Rev. 1)	Usage Inquiry System
SATN-10	01	June 78	(Rev. 2)	*SORTREQ*
SATN-11	01	January 76		*)RESET*
SATN-12	01	January 76		*)COPY*
SATN-13	10	March 78		Early Warnings
SATN-14	15	August 78	(Rev. 2)	Packages
SATN-15	15	April 76		Index
SATN-16	20	April 76		File System Must-Write Buffers
SATN-17	30	June 76		Formatting Primitive
SATN-18	01	July 76		*⎕FMT*
SATN-19	01	January 77		Fileprint
SATN-20	01	June 78	(Rev. 3)	System Variables and Functions
SATN-21	01	June 78	(Rev. 1)	*⎕WS* and *⎕FD*
SATN-22	15	January 77		APL Workspace Transfer
SATN-23	07	June 77		Comparison Tolerance
SATN-24	25	March 77		*)SYMBOLS*
SATN-25	15	May 77		Extensions to Argument Passing
SATN-26	10	September 77		Enhancements to the File System
SATN-27				
SATN-28	11	July 77		Terminal Control
SATN-29	15	June 78		System Time and Timestamps

COURSE SCHEDULE

SEMINARS:

EDMONTON	Time Series Forecasting (one day)	Oct 5	
TORONTO	Actuarial APL Techniques (1 day)	Oct 19	Dec 6
	AIDS - Introduction (2 days)	Oct 30	Dec 11
	Box-Jenkins (1 day)	Sep 8	Nov 14
	Data Base Design (1/2 day)	Sep 26	Nov 27
	Financial Analysis (1 day)	Sep 13	Nov 13
	Forecasting Methods (1 day)	Sep 11	Nov 16
	Graphics (1 day)	Oct 25	Dec 5
	MAGIC for Time Series Analysis (1 day)	Sep 12	Nov 9
	Plotting with SHARP APL (1 day)	Sep 21	Nov 21
	Regression Analysis (1 day)	Oct 24	Dec 13
	Report Formatting with SHARP APL (1/2 day)	Oct 11	Dec 7
	Saving Money with N-tasks + B-tasks (1 day)	Oct 26	Dec 14
U.K. (LONDON)	Appreciation of APL (1 day)	Oct 12	
	Systems Design (1 day)	Sep 7-8	Nov 16-17

INTRODUCTION TO APL

Dallas
(4 day)
Sep.5,7,12,14
Oct.3,5,10,12
Nov.7,9,14,16
Dec.5,7,12,14

Ottawa
(5 day)
September 11-15
October 2-6
November 6-10
December 4-8

U.K. London
(5 day)
Oct.18-20,23,24
Nov.8-10,13,14
Nov.29,30,Dec.1,4,5

Houston
(4 days)
September 4-8
October 2-6
November 6-10
December 4-8

Rochester
(5 days)
September 25-29
October 16-18
November 20-24
December 18-22

Vancouver
(3 days)
September 27-29
October 25-27
November 22-24

Minneapolis
(6 half-days)

Toronto
(3 day)
September 5-7
October 16-18
November 6-8
November 28-30
December 18-20

Winnipeg
October 10-13

INTERMEDIATE:
TORONTO (2 days) Sep 27-8 Nov 22-24
WINNIPEG Oct 24-27

ADVANCED:
TORONTO Advanced APL and Efficient Coding Techniques Nov 1-2
U.K. APL System Design Sep 7-8

SPECIAL COURSES:
TORONTO APL for Managers Oct 12-13
AMSTERDAM Introduction to APL, in Dutch Sep 20,21,22 Nov 22,23,24
DUESSELDORF/VIENNA/ZURICH, in German, scheduled on demand

UPDATE

☐ Please amend my mailing address as indicated.

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

☐ Send me a SHARP APL publications order form.

Name: _____

Co.: _____

Address: _____

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

International Branch Offices

Aberdeen
I.P. Sharp Associates Limited
5 Bon Accord Crescent
Aberdeen AB 1 2DH
Scotland
(0224) 25298

Amsterdam
Intersystems B.V.
Herengracht 244
1016 BT Amsterdam
The Netherlands
(020) 24 40 50
Telex: 18795 ITS NL

Atlanta
I.P. Sharp Associates, Inc.
Suite H-10
2550 Akers Mill Rd. N.W.
Atlanta, Georgia 30339
(404) 953-1020

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

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

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

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

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

Cleveland
I.P. Sharp Associates, Inc.
Suite 590, 27801 Euclid Ave.
Cleveland, Ohio 44132
(216) 261-0800

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

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

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

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

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

Houston
I.P. Sharp Associates, Inc.
Suite 925, One Corporate Square
2600 Southwest Freeway
Houston, Texas 77098
(713) 526-5275

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

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

Los Angeles
I.P. Sharp Associates, Inc.
Sherman Terrace Bldg.
18040 Sherman Way
Suite 118
Reseda, Ca. 91335
(213) 343-4617

Melbourne
I.P. Sharp Associates Pty. Ltd.
36 Elizabeth St.
South Yarra, Melbourne
Victoria, Australia 3141
(03) 244-417

Miami Lakes
I.P. Sharp Associates, Inc.
Suite D, Kennedy Bldg.
14560 N.W. 60th Avenue
Miami Lakes, Florida 33014
(305) 556-0577

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

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

New York City
I.P. Sharp Associates, Inc.
Suite 242, East Mezz.
200 Park Avenue
New York, N.Y. 10017
(212) 986-3366

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

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

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

Philadelphia
I.P. Sharp Associates, Inc.
Suite 407, 1420 Walnut Street
Philadelphia, PA. 19102
(215) 735-3327

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

San Francisco
I.P. Sharp Associates, Inc.
Suite C415, 900 North Point St.
San Francisco, Ca. 94109
(415) 673-4930

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

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

Sydney
I.P. Sharp Associates Pty. Ltd.
Suite 1342, 175 Pitt Street
Sydney, N.S.W., Australia 2000
(02) 232-5914

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

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

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

Vienna
I.P. Sharp Ges. mbH
Rechte Wienzeile 5/II
1040 Vienna, Austria
(222) 576-571

Warrington
I.P. Sharp Associates Limited
Paul House
89 - 91 Buttermarket St.
Warrington, Cheshire
England WA1 2NL
(0925) 50413/4

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

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

Zurich
I.P. Sharp A.G.
Badenerstrasse 141
8004 Zurich
Switzerland
(1) 241 52 42

SHARP APL Communications Network: Local Access Cities

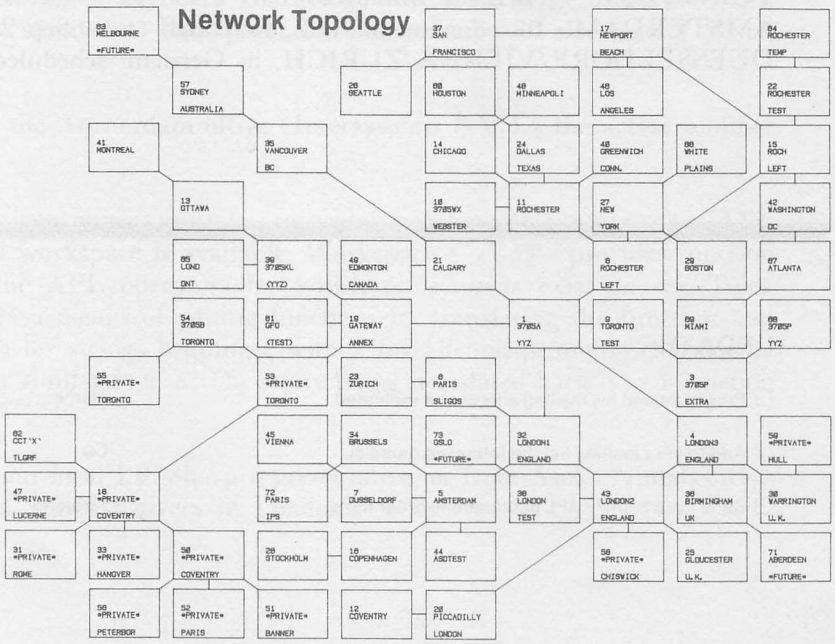
APL OPERATOR VOICE (416) 363-2051 ***COMMUNICATIONS (416) 363-1832***

Local dial access is available in all locations listed above. The SHARP APL Communications Network also provides local dial access in:

- Ann Arbor
- Buffalo
- Coventry
- Dayton
- Des Moines
- Detroit
- Ft. Lauderdale
- Greene (NY)
- Greenwich (Ct)
- Halifax
- Hamilton
- Kitchener
- Liverpool
- Manchester
- Milan
- Paris
- Raleigh
- Regina
- Saskatoon
- Syracuse
- White Plains (NY)

In the United States the SHARP APL Network is interconnected with the networks of TYMNET and TELENET to provide local dial access in more than 100 other cities.

Network Topology



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

Jeanne Gershater, *Editor* Ginger Kahn, *Assistant Editor*