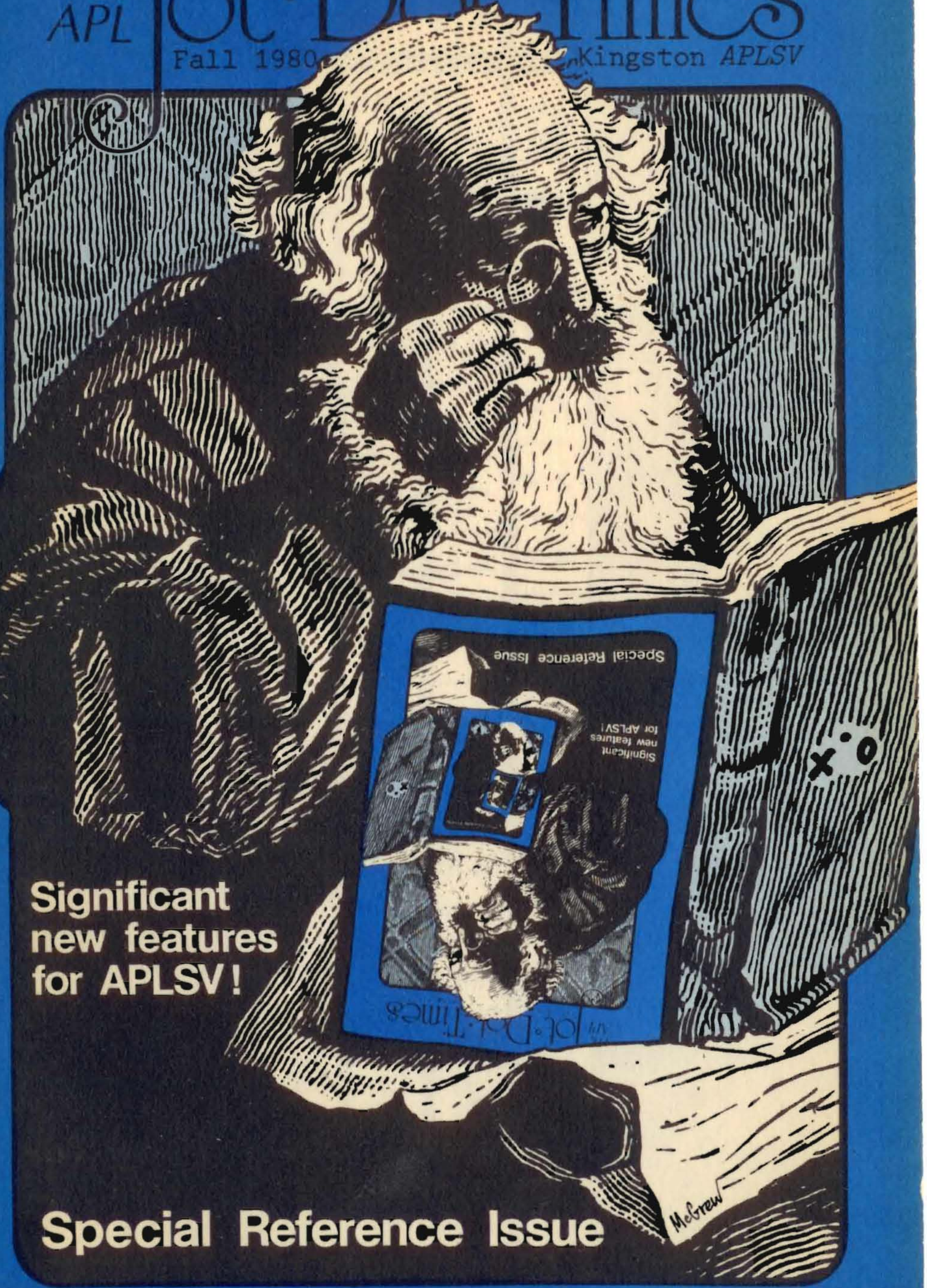
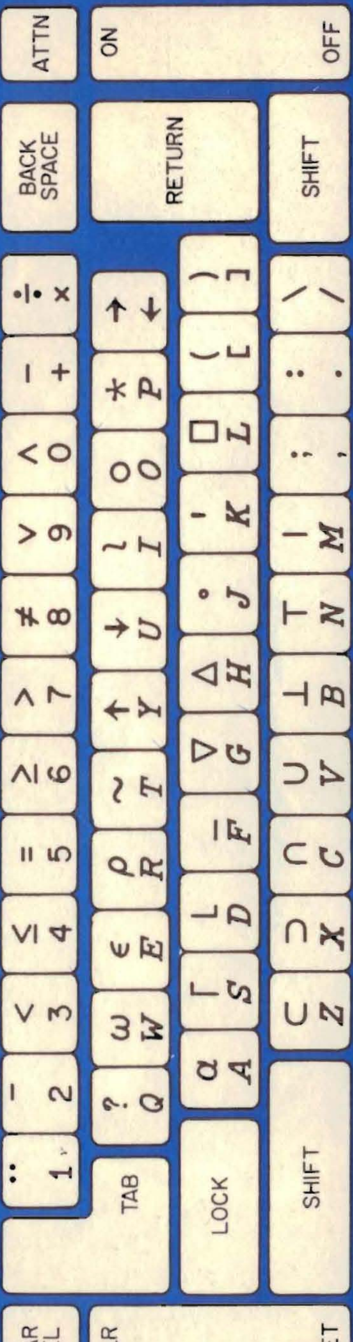
The APL Jot·Dot·Times

Fall 1980 ©   Kingston *APLSV*



Significant new features for APLSV!

Special Reference Issue


MAR REL
¨ 1
¯ 2
< 3
≤ 4
= 5
≥ 6
> 7
≠ 8
∨ 9
∧ 0
- +
÷ ×
BACK SPACE
ATTN
CLR
TAB
? Q
⍵ W
∊ E
⍴ R
~ T
↑ Y
↓ U
⍳ I
○ O
⋆ P
← →
RETURN
ON
LOCK
⍺ A
⌈ S
⌊ D
_ F
∇ G
∆ H
∘ J
' K
⎕ L
([
)]
SET
SHIFT
⊂ Z
⊃ X
∩ C
∪ V
⊥ B
⊤ N
| M
; ,
: .
\ /
SHIFT
OFF

The APL Jot∘Dot·Times

Some production notes

The text for this newsletter was entered on an *APL*-featured 3277 video terminal, and edited and composed by a set of *APL* functions called *The APL Text Machine*. The master pages were then typed by the computer on a mag-card terminal ("Cardinal"), using Scribe, Light Italic, and *APL* typing elements. The headings for the articles were then applied to the masters using commercially-available transfer lettering. The inter-leaving of dual fonts was handled by the Text Machine without the need for cut-and-paste.

While several people have suggested photo-composer methods for the text, it is important to us to keep the newsletter itself as an example of the type of document that any of our users could produce on their own subjects.

The illustrations scattered throughout the newsletter (including the cover) were created between 1850 and 1925, originally appearing in newspapers, advertisements, and periodicals.

Jon McGrew
Wordsmith

Cover Portrait: ...Could this be *APL*'s own Adin Falkoff, shown here trying desparately to make *some* sense out of the Jot∘Dot·Times' Picture Format article??

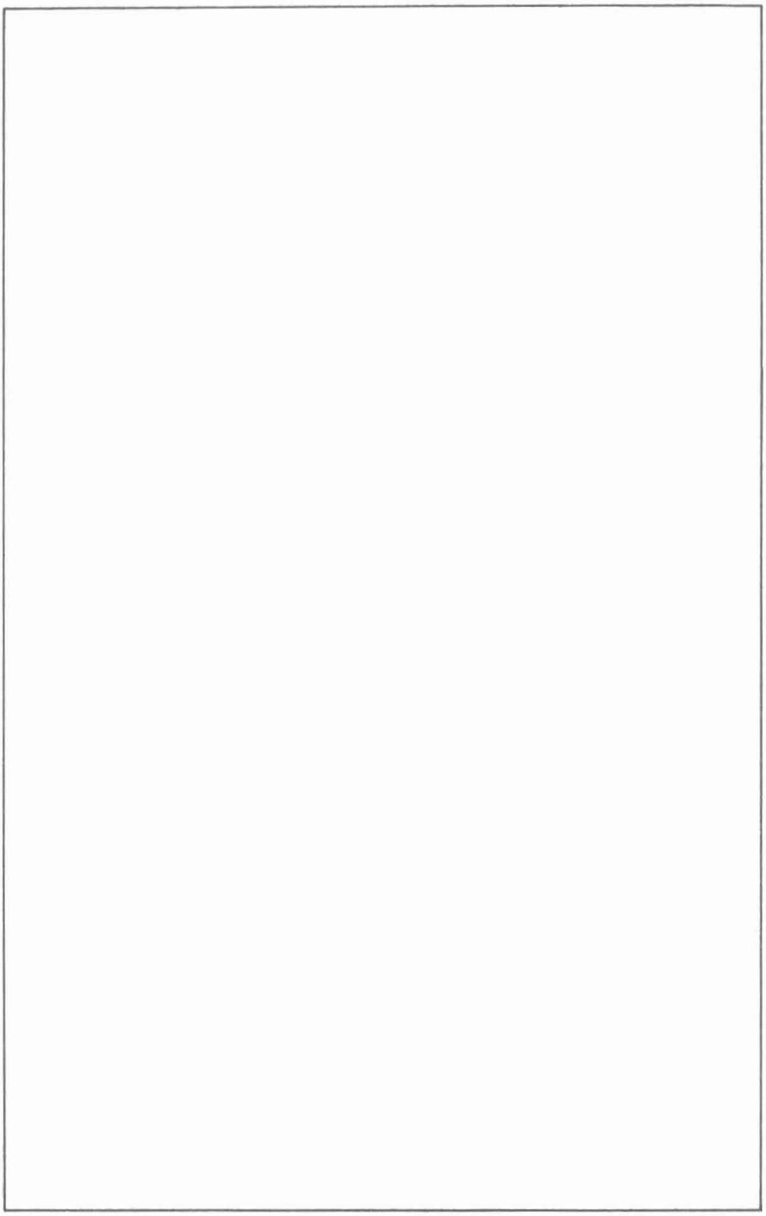


Figure 0: Clear Workspace

Preface

This is the Fall 1980 Edition of The Jot∘Dot·Times, published somewhat sporadically by the Kingston SCD *APLSV* Support Group. A copy is being sent to each of our customers.

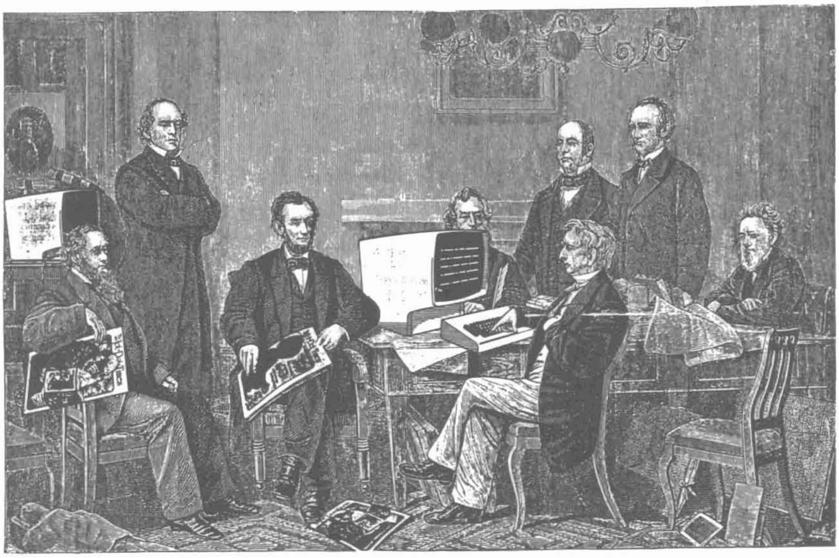
This particular issue has been in the works for a long time. We regret the delay in getting it to you, and we hope that you will find the features described here to be worth the wait.

We value your thoughts on this newsletter. If you have any comments or suggestions regarding either the newsletter or our service, please let us know. There is a Feedback form attached to the outside of the newsletter, and we have included some of the quotable quotes from the Feedback forms that we received from the previous newsletter.

Your comments can often help to mould future issues and future offerings on *APL*. Many of the facilities that we have offered in the past are a direct result of comments that were sent in on newsletter Feedback forms. Please, let's hear from you!

The Kingston *APLSV* Support Group

Table of Contents



Kingston APLSV Assistance Numbers

APL Hotline [Administration, Programming, Management, Operations, Line Services]	373-1234
APL Administration .... (Vince Dougherty and Millie Bartsch)	373-1234
System Status Recording	373-7817
Line Status Recording	373-4407

WHO'S WHO in Kingston *APL*

Interactive Terminal Systems Manager - Bill Davis	373-4652
ITS Secretary - Rhonda Johnson	373-1239
Planning and User Support Manager - Chuck Norcutt	373-2471
APL User-Support Programmer - Rich Euler	373-6216
APL User-Support Programmer - Mike Harelick	373-2405
APL User-Support Programmer - Mike Higgs	373-2254
APL User-Support Programmer - Jon McGrew	373-2538
APL User-Support Programmer - Joe Traina	373-1365
System Support Manager - John Brink	373-4294
MVS/JES3 System Programmer - Ken Jonas	373-1198
MVS/JES3 System Programmer - John Opalach	373-6775
APL System Programmer - Mike Van Der Meulen	373-1082
Administration & Operations Manager - Tom Dederick ...	373-6413
Console Operators	373-6661
Control Centre (ws & file transfers) - Rich Dill ...	373-6772
APL Administrator - Vince Dougherty	373-1234
APL Administrator - Millie Bartsch	373-6536 or -1234
2nd shift Operations Manager - Skip Frasier	373-6966
Console Operator - on rotation	373-6661 or -6837
3rd shift Operations Manager - Don Miller	373-6966
Console Operator - Hank Adams	373-6661 or -6837
Manager of the Tape Library Walt Hackett	373-7343
Tape Library - Morgan Moore	373-6673
Network Services Manager - Ivan Pece	373-7839
Telephone line problems - Wilma Quick	373-7891
Time Accounting Manager - Ed Goodman	373-4049
Billing information - Marguerite Lasher	373-4222
Billing information - John Offermann	373-2491

Introduction

This issue of the Jot∘Dot Times is our Special Reference Issue. So what's that? Well, we have discovered that once we have studied the *APL* manuals, we kind of understand what's going on, so we don't really have to carry a stack of manuals with us all of the time. But *no way* are we *ever* going to memorize all of the reference tables in those manuals! So, here we are, still carrying all of those manuals around. It would be nice (sez us) if we could gather together all of the commonly-needed charts and tables into one convenient handbook, and keep that around. Aha... lucky us; here it is. [applause]

Publishing all of these learned-looking tables also gives us an academic air; perhaps some folks will even be deluded into thinking that we wrote them. Better and better. Besides, you see, what we have here is clearly a most impressive piece of technical research and compilation (as opposed to the shoddy piece of plagiarism that you probably first assumed it to be).

True, the text is ours. There's no escaping that admission. However, if the truth be known (and it probably will be), we swiped the tables. But what the hey, you know? "The Knowledge of The Universe Can Be Yours, Through Plagiarism." [You can quote us on that.] Most of them came from The *APL* Language Manual (GC26-3847) and The *APLSV* Version 3 User's Guide (SH20-9087). The tables have, in many instances, been modified so that they more properly reflect local conventions on Kingston *APLSV*; they therefore do not necessarily show a true picture of life on other systems. Other manuals are recommended on the "General Information" page; all of them can be ordered internally from Mechanicsburg. If you don't already have each of these manuals, you should order them. They are really quite helpful. While this issue will hopefully mean that you won't have to carry the manuals around all of the time, everyone needs to refer to the text sometimes, and the manuals are still very good reference manuals to have available.

As this issue is being published, quite a few new system enhancements are being installed. It's always quite a pleasure to be able to report the new goodies as they become available. There have been on-going enhancements in the past, both from the *APLSV* Central Support Group and from The *APL* Design Group, but most of what has been released in the past couple of years hasn't been visible to you as a user... these changes have been system speed-ups and measurement tools for those of us that supply the *APL* service. You may not have noticed those changes, although with the increased *APL* workloads you would have noticed the lack of them if they hadn't been installed. But now we have what we expect will be some very popular new system changes, all quite visible.

Jon

o.x

Operating Schedule for Kingston APLSV

```
  Sunday       Monday      Tuesday    Wednesday   Thursday     Friday      Saturday
|----------|-----------|-----------|-----------|-----------|-----------|-----------|
□□□□****□□□□□⊞⊞⊞⊞⊞⊞⊞⊞⊞⊞ ⊞⊞⊞⊞⊞⊞⊞⊞⊞⊞⊞ ⊞⊞⊞⊞⊞⊞⊞⊞⊞⊞⊞ ⊞⊞⊞⊞⊞⊞⊞⊞⊞⊞⊞ ⊞⊞⊞⊞⊞⊞⊞⊞⊞⊞⊞⊞⊞   ⊞⊞⊞⊞□□□□
                       ↑           ↑           ↑           ↑             ↑       ↑
Downtimes:             12-2am      12-2am      12-2am      12-2am        2-7am   |
---------                                                                        |
   Time is shown as:   ⊞↔Up and attended                                         |
                       □↔Up, but unattended . . . . . . . . . . . . . . . . 4:30pm
                       *↔On the 2nd Sunday of each month, APL will
                          be down for maintenance, typically 8am→5pm
```

--- Changes to this schedule will be broadcast at least one day in advance ---

Please note that this schedule is the normal weekly schedule for the Kingston *APLSV* systems... it does not take into account any special scheduling due to holidays, etc. For schedule exceptions, ")*LOAD* 1 *NEWS*" and type "*SCHEDULE*".

System configuration

Here's a brief run-down of our current system configuration. This list will be periodically updated as things change... ")*LOAD* 1 *SYSTEM*" for a current listing. At publication time, the listing looks like this:

```
IBM Corporation, SCD, Kingston, NY

Host: MVS/JES3, Rel 3.8
APL:  APLSV IC4 IR3
      [Internal Consolidation Release 4, Incremental Release 3]
      (Program No 5799-AQC, modified for internal use)
TSIO: IC4 IR3
                      System N       System H       System L
                      ----------     ----------     ----------

CPU ................. Sys/370 168    Sys/370 168    Sys/370 168
Main storage ........ 8 megabytes    8 megabytes    8 megabytes
Jobs on System ...... APL (100%)     APL (100%)     APL    (75%)
                                                    TSM    (10%)
                                                    Batch  (15%)

All workspaces and files are common to all three systems.

Workspace size:
  [measured as ⎕WA after )CLEAR]:  169,276 bytes
  [maximum overall size]:  173,376 bytes

Permanent storage:
  APL:  3330 Mod 11 disks
  TSIO: 3330 Mod 11 disks

Terminals supported:
                              Dial-up
                        ←-Start/Stop-→  BSC/SDLC (as appropriate)
  Line speed:           134.5 300 1200  4800-baud
                        ----- --- ----  ----

  2741, 1651, MCST:       ×
  5100, 5110, 5120:       ×     ×
  3767:                   ×     ×
  Tektronix 4013/4015:    ×     ×    ×
  Most 32xx devices:                        ×  ←------[see page 16]
  OEM terminals with\
   Correspondence or >    ×     ×    ×
   BCD line codes    /

APL print train:
  APLFULL

APL usage rates for 1980:
  Connect time:  $  4.25/hour
  CPU time:      $ 21.00/CPU minute
  Storage:       $   .02/track/week  [1 track=13,030 bytes]
  TSIO surcharge: none (all users are given TSIO capability)
```

General information

```
APL typeball (Correspondence) ............ Part Number 1167987
             (BCD/EBCD) .................. Part Number 1167988

APL keyboard stickers .................. Form Number GX20-1783
```

* * *

The following manuals may be ordered from the Mechanicsburg Distribution Centre:

GH20-0689	*APL*\360 Primer [old but good beginner's manual]
SH20-9087	*APLSV* Version 3 User's Guide [reference manual]
GC26-3847	*APL* Language [reference manual]
SR20-7183	*APL* - An Introduction (textbook) [lessons]
G320-6103	*APL* Programming Guide [a "how-to" reference]
S320-5996	The *APL* Handbook of Techniques [reference]

* * *

For information about our system, we recommend the following workspaces:

```
)LOAD 1 NEWS ...... General info, system schedule; should be
                    checked DAILY for news of system changes
)LOAD 1 ACCOUNT ... Billing and quota information
)LOAD 1 AIDS ...... Contains "ALLOCATE"... the only way to
                    create a permanent dataset, and other
                    general aids.
)LOAD 1 CATALOG ... Locations of Public Library info
)LOAD 1 FILELIB ... Info about your files (if any)
)LOAD 1 FORMS ..... Forms for requesting a new APL account,
                    or for changing the billing/mailing info
                    for an existing account
)LOAD 1 PHONES .... A complete and current list of phone
                    numbers that you can use for signing on
)LOAD 1 SYSTEM .... Technical data about our system
```

Typing "*DESCRIBE*" after loading each workspace will give you complete information on how to use that workspace.

For *APL* self-education, we recommend:

```
)LOAD 45 INDEX .... Library 45 contains a package of 54 APL
                    lessons.  This workspace gives a course
                    description.
)LOAD 45 LESSON1 .. The first of 54 APL lessons
```

New facility for tracking file usage

A new Public Library workspace is now available for tracking the usage of your datasets: *)LOAD* 1 *SMF*. *APL* has for some time kept track of who accesses each dataset; this is gathered as SMF (System Monitoring Facility) Data. We have had requests in the past to recover information about who was using a particular file, often to help in tracing a suspected security breach. This data is now easily available to all of the users of Kingston *APLSV*. This workspace retrieves data showing a summary of accesses to each of your files during the past week.

Its usage looks like this:

```
      )LOAD 1 SMF
SAVED  16.11.05 08/27/80

      SMF
***FILE DATE IS 08/23/80 THRU 08/29/80

      SHOW DSN 'MYWORK'
   ACCOUNT ACC     EXCP        BYTES  DSN

      1234   9       27      351,810  12345 MYWORK
      6789  28    1,221    7,871,787  12345 MYWORK
      9876  16       50      322,350  12345 MYWORK
     24680   4       32      416,960  12345 MYWORK
```

The data is summarized by filename and accessing account number. No attempt is made to show every access to your files, minute by minute, since the resultant listing could be very cumbersome. And since most users need just the summary, we have done that part for you. If you *DO* need the detailed entries to track a suspected security problem, we can get that data for you.

Some warnings and disclaimers

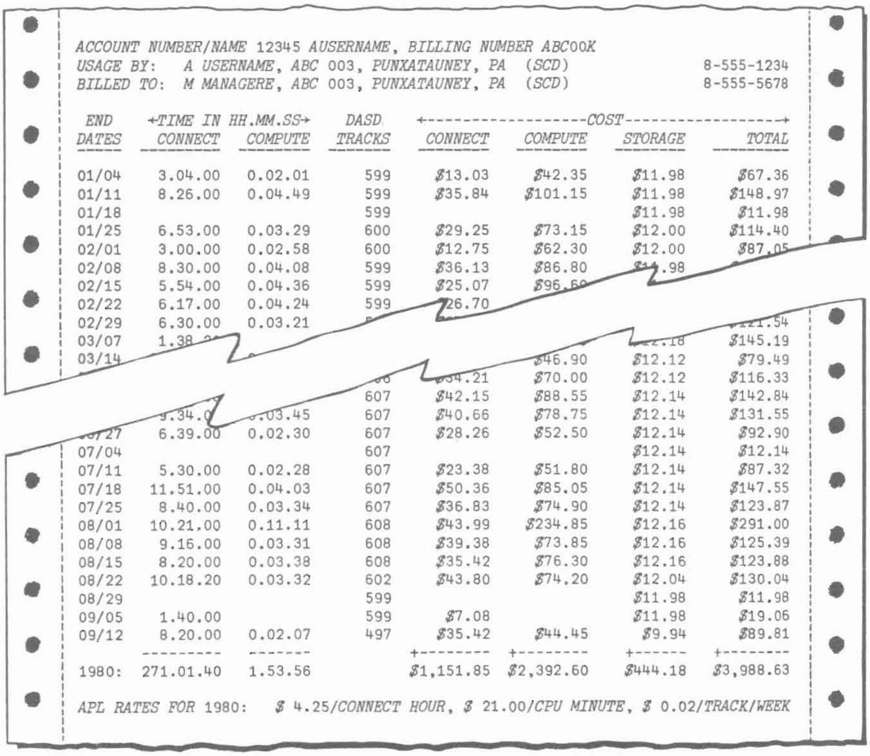
If you suspect a breach in the security of your application, or have any other need to pursue this information, make sure that you follow through on it during the week of the access in question... the data is replaced each Saturday. Due to the volume of the data, historic records are not kept on-line.

Next, the inevitable question of "who are all those people?". This can get sticky. As a general rule, we will NOT match up account numbers and names; that's considered to be quite privileged information, since it could be misused to gain access to another user's workspaces and datasets. However, we recognize the fact that some real security questions may come up on occasion that require such data. Therefore, any such request will be handled by management on an individual basis.

New facility for tracking your billing

A new workspace is now available for retrieving historical billing information for your account. Workspace 1 *BILLING* contains data for both this year and last year, and is updated weekly as the billing is run.

The functions in the workspace extract the data from our history files and allow you to generate your own reports, or if you wish to use our standard report format, the reports will look like this:



```
ACCOUNT NUMBER/NAME 12345 AUSERNAME, BILLING NUMBER ABC00K
USAGE BY:   A USERNAME, ABC 003, PUNXATAUNEY, PA  (SCD)                 8-555-1234
BILLED TO:  M MANAGERE, ABC 003, PUNXATAUNEY, PA  (SCD)                 8-555-5678

 END     ←TIME IN HH.MM.SS→    DASD    ←-----------------COST-----------------→
DATES    CONNECT    COMPUTE   TRACKS    CONNECT    COMPUTE    STORAGE      TOTAL
-----   ---------   -------   ------   --------   --------   -------   --------

01/04     3.04.00   0.02.01      599     $13.03     $42.35    $11.98     $67.36
01/11     8.26.00   0.04.49      599     $35.84    $101.15    $11.98    $148.97
01/18                            599                          $11.98     $11.98
01/25     6.53.00   0.03.29      600     $29.25     $73.15    $12.00    $114.40
02/01     3.00.00   0.02.58      600     $12.75     $62.30    $12.00     $87.05
02/08     8.30.00   0.04.08      599     $36.13     $86.80       .98
02/15     5.54.00   0.04.36      599     $25.07     $96.60
02/22     6.17.00   0.04.24      599      26.70
02/29     6.30.00   0.03.21                                                 .54
03/07     1.38                                                   .18    $145.19
03/14                                               $46.90    $12.12     $79.49
                                            .21     $70.00    $12.12    $116.33
                                 607     $42.15     $88.55    $12.14    $142.84
           .34.0       .03.45    607     $40.66     $78.75    $12.14    $131.55
   27     6.39.00   0.02.30      607     $28.26     $52.50    $12.14     $92.90
07/04                            607                          $12.14     $12.14
07/11     5.30.00   0.02.28      607     $23.38     $51.80    $12.14     $87.32
07/18    11.51.00   0.04.03      607     $50.36     $85.05    $12.14    $147.55
07/25     8.40.00   0.03.34      607     $36.83     $74.90    $12.14    $123.87
08/01    10.21.00   0.11.11      608     $43.99    $234.85    $12.16    $291.00
08/08     9.16.00   0.03.31      608     $39.38     $73.85    $12.16    $125.39
08/15     8.20.00   0.03.38      608     $35.42     $76.30    $12.16    $123.88
08/22    10.18.20   0.03.32      602     $43.80     $74.20    $12.04    $130.04
08/29                            599                          $11.98     $11.98
09/05     1.40.00                599      $7.08               $11.98     $19.06
09/12     8.20.00   0.02.07      497     $35.42     $44.45     $9.94     $89.81
        ---------   -------            +--------  +--------  +------   +--------
1980:   271.01.40   1.53.56           $1,151.85  $2,392.60   $444.18  $3,988.63

APL RATES FOR 1980:   $ 4.25/CONNECT HOUR, $ 21.00/CPU MINUTE, $ 0.02/TRACK/WEEK
```

The information provided from this workspace is from the same files that are used to produce the *APL* Monthly Utilization Reports, which are mailed to each billing manager every month. Those reports will continue to be sent out, since part of their *raison d'etre* is to keep the billing managers attuned to any abnormalities in the billing that they might not otherwise have noticed or checked (both for budgeting and security reasons).

Bigger workspaces

As you have undoubtedly already noticed by this time, the workspace size for all Kingston *APLSV* users has been increased from 125,932 bytes to 169,276 bytes of user area, as measured by ⎕*WA* after)*CLEAR* [173,376 bytes overall]. You will still have the same number of workspaces in your quota, giving you one-third more space (or, over twice the space that was available four years ago).

If you have an application which checks ⎕*WA* to determine how much data may be processed at one time, substantial improvements in performance could result automatically.

There is one concern which should be considered: moving an application to another *APL* system will require that you do not store more material in a workspace than the receiving system is prepared to accept. Our new workspaces can be transferred to other *APLSV* systems exactly as before, providing that you have not saved more material than they can fit in their workspaces. And remember, the material not only has to <u>fit</u> in the workspace, it has to <u>run</u> in the workspace. Contact us in *APL* support if you have any questions on considerations such as this.

Kingston APLSV Printer Output Classes

Class	Print	←--------(H×W)--------→			
		←-Page Size-→		Chars/	
[*SYSOUT*=_]	Train	Inches	Chars	Inch	Type of Output
P	*APLFULL*	11×14	66×130	6×10	Normal *APL* output
I	*APLFULL*	11×14	66×130	6×10	IBM Confidential
A	GF12	8.5×14	66×156	8×12	Fastest output
B	------	------	80 col	----	Punched cards
C	GF12	8.5×14	66×156	8×12	IBM Confidential
N	*APLFULL*	11×8.5	66×80	6×10	All-white paper
K	TN	1.3×4	7×36	6×10	Labels
4	ST12	8.5×14	66×156	8×12	Text output

Description of Print Trains or Character Sets	
APLFULL	Contains all of the characters that appear on the *APL* typeball, plus upright (non-italicized) caps and lower-case characters: **ABCDEFG abcdefg *ABCDEFG* α⊥∩⌊∊_∇ 0123456**
GF12	"Gothic Folded, 12 pitch" ...3800 laser printer, upper-case only; no lower-case, no *APL*: ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 (!& <)=*>¬$
ST12	"Serif Text, 12 pitch" ...3800 laser printer, caps and lower-case; no *APL*: ABCDEFGHIJKLMnopqrstuvwxyz0123456789 < = >▪}
TN	Text train: caps and lower-case; no *APL*: ABCDEFGHIJKLMnopqrstuvwxyz0123456789

SYSOUT=P and *SYSOUT=I* are the two normal output classes for Kingston *APLSV*. This output will be printed on the all-white side of green-striped paper.

The use of workspace 31 *PRINT* is recommended for printing data.

The page height is normally indicated as being 66 lines; this includes top and bottom margins. Skipping to a new page (skip to channel 1) will result in the print train being positioned about three-quarters of an inch from the top edge of the paper, leaving about 61 printed lines to the bottom edge of the page, or, comfortably, about 55 printed lines with proper margins. Please do not ask the operators to reposition the paper for special requirements. The 3800 cannot print over page perforations (based on its hardware design).

Punched output will not be interpreted (the data will not be printed on the cards).

Labels (*SYSOUT=K*) will stay in alignment if a skip to a new page is specified for each new label. Better registration can result if you start the output with about 25 alignment labels.

New keywords for TSIO

[Kingston only]

COPIES

A new keyword is now available for use in conjunction with the *SYSOUT* keyword: *COPIES*=n, where "n" is an integer value from 0 through 255.

For example, rather than sending a report to the system printer twice in order to get two copies, simply modify the appropriate TSIO *SYSOUT* command so that it is in the form "*SW SYSOUT=P*,*COPIES*=2,*BLKSIZE*= ...".

A value of 0 may be useful for debugging programs without generating any output.

The use of this parameter is, of course, optional. The default is *COPIES*=1, so it certainly doesn't need to be specified for that.

SC

This keyword over-rides the security classification normally associated with any of the output classes:

SC=UN	Unclassified	Printed on outside
SC=IU	IBM Internal Use [default]	header pages only;
SC=IC	IBM Confidential	inside pages are
SC=IR	IBM Confidential Restricted	always up to you.

SI

This keyword indicates that "special instructions" are needed for the output. Valid entries are *SI=H* (hold for pickup), or *SI=S* (special). To enter the special instructions for this second case, ")*LOAD* 1 *AIDS*" and type "*SYSOUT*". This is the only time that this function needs to be used.

USER

Printed output from our machine room now has your name and address printed on it, based upon the address that we have on file for you (need to change it? ...mail the Change Form from Page 92). The "*SYSOUT*" function in 1 *AIDS* no longer needs to be used for every report. The *USER* keyword specifies the numeric sign-on number of another user of the Kingston *APL* system to which the output should be mailed. If the serial number is that of the requestor, it is ignored. Enter this keyword in the form *USER*=12345.

Note: The *SI* and *USER* keywords are mutually exclusive; if both are entered, the command will be rejected as an *IMPARSIBLE COMMAND*.

* * *

Also now supported on TSIO is a *RLSE* ("release") parameter for use with the *SPACE* keyword. This allows you to release unused space when the dataset is closed following its initial creation. Refer to Figure UG8 in the back of the newsletter.

Symbolic parameters in indirect commands

First, some background: "What's an indirect command?".

A reserved dataset is a dataset that can be accessed only by its owner... if you are signed on with any account number other than the account number that the reserved dataset is under, an attempt to access it will result in a return code of 2 - *RESTRICTED COMMAND*. A reserved dataset is distinguished from a standard (non-reserved) dataset by means of the account number: for a non-reserved dataset, it's a positive number that matches the account number; for a reserved dataset, it's the negative version of that same account number. For example, a non-reserved dataset name is "12345 *MYDATA*". A reserved dataset name looks like this: "¯12345 *MYDATA*". If you don't specifiy an account number at all, it defaults to a non-reserved dataset (anyone can access it, if they know the name).

Well, a reserved dataset seems to be a nice security feature, but there's one major problem: if it keeps everyone except the owner out, that's somewhat overly restrictive. All of the processing for any sensitive project would have to take place on a single account number. So there's an extension to that.

A command dataset is a special instance of a reserved dataset; that is, a command dataset is always reserved, but further, it's a special dataset which contains (you guessed it) commands and lists of user numbers that can use each of the commands. [See the function called "*IC*" in workspace 1 *AIDS* for building and maintaining command datasets.] You can put any TSIO command that you wish in your command dataset, and may specify exactly who may use each of the commands. When they use them, they are executing them just as if they were under your account number; that is, they have your access to those commands, and so can access your reserved datasets just as if they were you... if you authorize them.

If you are user 12345, you'd get this response:

```
            CTL←'SR DSN=¯12345 MYDATA'
            CTL
      0        ←-- (successful)
```

But if you are any other user, you'd get this:

```
            CTL←'SR DSN=¯12345 MYDATA'
            CTL
      2 1      ←-- (restricted command)
```

No problem. If you want to let a certain user (or even all users) have a specific type of access to your dataset, you can put that same command into a command dataset, and authorize the appropriate users. Then, they would access it like this:

```
               CTL←'IC DSN=¯12345 GATE(7)'
               CTL
         0        ←-- (successful)
```

In this command, the name of the command dataset is "¯12345 *GATE*", and the command that this user is authorized to access has been put into command slot 7 of that dataset. The command that's being used is *IC*... "Indirect Command".

So far so good. What more could you ask for? Well, just one more problem: the command that appears in the command dataset had to be a <u>complete</u> command, not just a portion of the command. Therefore, if you wanted to let certain people access any of your twenty datasets for indexed-read only, you had to have twenty commands. Hmmm, seems like there ought to be a better way....

Introducing Symbolic Parameters

Now there is a better way: a <u>symbolic parameter</u> can be thought of as being, in many respects, analogous to an *APL* variable. What's often needed is a way to put <u>most</u> of the command in the dataset, and stipulate what parameters may be added, but then allow the user to supply values for those parameters.

A symbolic parameter is distinguished from other TSIO terms and values in that it begins with "∧" (the *APL* version of what JCL sees as an ampersand: "&"). This term would then be used both in the command within the command dataset and in the command that calls that one. For example, assume the command in the command dataset to be "*SR DSN*=¯12345 *ABC*,*DISP*=∧*D*". When this command is invoked, the command would look like this:

```
               CTL←'IC DSN=¯12345 GATE(7),∧D=SHR'
               CTL
         0
```

Notice that the "*DISP*=∧*D*" in the command within the command dataset will be filled in with the value supplied by the user, "∧*D*=*SHR*", allowing TSIO to read the string as "*DISP*=*SHR*".

If you are a user who is entering the *IC* command, the name of the symbolic parameter that you enter must, of course, match the name that appears in the command dataset. That name can be from 1 to 8 characters, alphanumeric (*A*→*Z*, 0→9). The first character may not be numeric. The first three characters may not be *SYS*... that's a reserved prefix, which gives us some additional features.

Reserved Names for Symbolic Parameters (*∧SYS*...)

Several special names have been established which the system will replace with a value upon their use (a table of these terms follows shortly). For example, one such term is *∧SYSDATE*. This term returns the current date in the form "YYDDD" (Julian date). If you wish to be able to create a new dataset whose name contains the current date, this can be done by placing a command in a command dataset like this:

SW DSN=¯12345 *D∧SYSDATE*

If this were the 250th day of 1980, the system would then treat the command as though it were

SW DSN=¯12345 *D*80250

Notice that we put an alphabetic character ("*D*") in the command so that the dataset name won't start with a numeric. The symbolic name may be used to replace any <u>portion</u> of a term (but it <u>can't</u> be used to pass more than one term). For example, consider the following commands:

Command dataset contains:	User enters command as:	TSIO sees it as:
IR DSN=¯12345 *D∧SYSDATE*	*IC DSN=*¯12345 *GATE*(7)	*IR DSN=*¯12345 *D*80250
I∧X DSN=¯12345 *MINE*	*IC DSN=*¯12345 *GATE*(8),*∧X=RW*	*IRW DSN=*¯12345 *MINE*
SR DSN←¯12345 *WEEK∧WK*	*IC DSN=*¯12345 *GATE*(9),*∧WK=*19.*B*	*SR DSN=*¯12345 *WEEK*19.*B*
SR DISP←∧D,DSN←¯12345 *ABC*	*IC DSN=*¯12345 *GATE*(12)	*SR DISP=,DSN=*¯12345 *ABC*

...Trivia Department: Notice that a specification arrow (←) can always be used in place of an equal-sign in any TSIO command... TSIO will make the substitution.

Notice also that the command dataset won't see any value from a symbolic name if that name isn't specified; that's okay, the system will supply the default value [see the table of default values on page 86]. In the example used here, TSIO would treat the command as though it had been "*SR DISP=SHR,DSN=*¯12345 *ABC*".

If you would prefer to specify your own defaults, you can do that: just enter the command in the *IC* dataset like this: *BLKSIZE←∧BLK,∧BLK←*13030.

Reserved Names for Symbolic Parameters

Term	Purpose	Sample
∧*SYSACCT*	Account number	12345
∧*SYSPACCT*	Positive account number (same as above)	12345
∧*SYSNACCT*	Negative account number	¯12345
∧*SYSPCODE*	"Scramble" of positive account number	*AAAADADJ*
∧*SYSNCODE*	"Scramble" of negative account number	*PPPPMPMH*
∧*SYSTIME*	Time of day (HHMMSS)	152148
∧*SYSHOUR*	Hour (HH)	15
∧*SYSMIN*	Minute (MM)	21
∧*SYSSEC*	Second (SS)	48
∧*SYSDATE*	Date (YYDDD)	80245
∧*SYSYEAR*	Year (YY)	80
∧*SYSJDAY*	Day of year [Julian day] (DDD)	245

Note: "∧*SYS*" is a reserved prefix; no user-generated symbolic names may begin with "∧*SYS*". Also, no specification is allowed to these names.

These names (as all symbolic parameters) may only be used by a command within a command dataset (they may not be invoked directly).

The "scramble" of the account number is an encoding that TSIO uses to name datasets. Although you see your dataset as "12345 *MYDATA*", TSIO sees it as "*TSIO.AAAADADJ.MYDATA*". [While you (as a standard Level 10 user) can't specify this name directly through TSIO, knowledge of its existence may be helpful for dealing with datasets moving on and off the system.] All TSIO datasets start with "*TSIO.*", to identify them, and the next eight characters are an encoding (or "scramble") of the account number: 12345↔*AAAADADJ*, and ¯12345↔*PPPPMPMH*. The functions for performing this translation are as follows:

```
     ∇ Z←SCRAMBLE N
[1]    Z←⍉'ABCDEFGHIJKLMNOP'[⎕IO+(8⍴16)⊤N]
     ∇
       SCRAMBLE 12345
AAAADADJ

     ∇ Z←UNSCRAMBLE N;⎕IO
[1]    ⎕IO←0
[2]    Z←N
[3]    →(~∧/Z∊' ABCDEFGHIJKLMNOP')/0
[4]    →(8≠⍴Z)/0
[5]    Z←16⊥'ABCDEFGHIJKLMNOP'⍳Z
[6]    Z←Z-4294967296×Z>2147483647
     ∇
       UNSCRAMBLE 'AAAADADJ'
12345
```

New system command:)PASSWORD

As you perhaps already discovered from the news item in 1 *NEWS*, the old method of changing your password at sign-off time [*)OFF*:newpass] is no longer supported. Instead, there is now a new system command: *)PASSWORD*.

Used without an argument, it returns the date by which the current password must be changed:

```
      )PASSWORD
EXPIRES 09/02/80
```

To change your password, you must supply both the old password and the new password that you wish to start using:

```
      )PASSWORD oldpass:newpass
EXPIRES 10/03/80
```

The date that is returned following that operation is the expiration date of the new password.

If the "old" password that you enter doesn't match the one that's currently in use on your account, you will get a response of "*OLD PASSWORD INCORRECT*". You could also get a message of "*NEW PASSWORD UNACCEPTABLE*", usually due to the proposed new password being too short. In order to be in compliance with Corporate Security, a new password on the Kingston system must meet these length requirements:

Mixed alphabetic and numeric password	4→8 elements
All alphabetic password	4→8 elements *
All numeric password	6→8 elements

(Alphabetics are *A*→*Z* and *A̲*→*Z̲*; numerics are 0→9)

* * *

As before, passwords may be up to eight characters in length... if a longer one is mnemonically meaningful to you, it's usable, but only the first eight characters will be examined.

When you sign on to *APL*, the message that used to remind you "*PASSWORD LAST CHANGED* ..." now reads "*PASSWORD EXPIRES* ...".

Another change involves the action that comes about if you don't change your password within the prescribed time. Accounts used to be locked out of the system for stale passwords, giving a message of "*NUMBER LOCKED OUT*" when a sign-on is attempted. Now you will be greeted with a message saying "*PASSWORD EXPIRED*". So what's the difference? Well, you can't sign on to the account under either case, but if the

* Yes, the eagle-eyed among you may think that these figures appear to be in conflict with Corporate Security Instruction 104; however, they were chosen through agreement with Security, based upon the large size of *APL*'s alphabetic character set.

account is <u>locked</u>, you also can't load its workspaces from another account. If the account is being prevented from use because of a stale password, you <u>can</u> use the workspaces from other sign-ons.

If you <u>do</u> let your password lapse, don't worry; assuming that you call within a reasonable time after the password expires, it can still be resolved quickly. A phone call to Kingston *APLSV* Administration is all that's needed... they're on T/L 373-1234. They will want to know only your sign-on number, not your password. They can then assign a "grace" period for signing on with the old password, so that you can get on and change it.

If you don't call Administration within another 30 days after your password lapses, the account will be <u>locked</u> (you can't sign on, and no one else can use the workspaces). If you still don't contact Administration within 30 days after that, they will have to assume that you've gone away, and will archive the account to tape, where the workspaces will be held for two years. If your account has been deleted in this manner, reinstatement to the system will require filling out a new application form.

Password change requirements on Kingston *APLSV*

```
  Last
Password                   (63 days
 Change                     maximum)
   |       Recommended         |                       Eligible *
   |      next password        ↓                         for
   |         change         INHIBIT       LOCK        DELETION
----φ===========φ===========φ===========φ===========φ---------→
   |         1 month       2 months     3 months     4 months
                               |            |            |
                               |            |            |
After these points, \     PASSWORD       NUMBER       NUMBER
 a sign-on attempt   >     EXPIRED       LOCKED       NOT IN
  will result in:   /          |           OUT        SYSTEM
                               |            |            |
                               |            |            |
User can sign on normally--→   |            |            |
                                            |            |←---------→
                                            |            | User must
Wss can be loaded from another account--→  |            |  re-apply
                                                         |
-------Account is still on the system, and --------→    |
user can usually be re-instated with one phone call→    |
```

* The actual deletion takes place after the next quarterly back-up tapes are made, to insure that a suitable two-year retention copy of all of the workspaces will be available.

Coming soon!

Improved standard 3270 support

In order to give you additional flexibility and support for more terminal types, the *APLSV* Central Support Group is in the process of modifying the software that controls the 3270s so that it will run under "VTAM" support (that's Virtual Telecommunications Access Method). So how does this help you, the terminal user? ...glad you asked....

Currently, a 3270 has to be wired directly into the *APL* system, with each terminal tying up precious resources in the machine even when it's not in use... and the connection can't be used by anyone else, even if you're not using your terminal. Under the VTAM scheme, each 3270 will be consuming resources only when it's active. Also, where you used to be connected to only one *APL* system, you will be able to select any of the three systems through VTAM... a definite plus should one of the three systems go on the blink. For times when you don't care which system you use, the eventual plan is to let VTAM automatically select the "best" system for you to sign on to, based upon its current performance.

Under the new release, additional types of 3270 terminals will be supported. These are:

Device type	Supported models
3276 Control Unit/Display	Models 2, 3 and 4
3277 Display Station	Model 2
3278 Display Station	Models 2, 3, 4 and 5
3279 Display Station	Models 2, 3 and 4
3284 Printer	Models 1 and 2
3286 Printer	Models 1 and 2
3287 Printer	Models 1 and 2
3288 Printer	As 3286 printer (*APL* not available)
3289 Printer	Models 1 and 2 (*APL* not available)

The following differences may be noted when using a 3270:

1. The user must sign on to *APL* within two minutes of either the initial display of the logo or signing off *APL*. If the sign-on is not completed within two minutes, the device will be returned to VTAM.

2. If the user is not siged on to *APL*, the PA2 attention key will immediately return the device to VTAM.

3. Through VTAM there are many 3270 devices competing for a specific number of *APL* ports; therefore, the user could receive the message *NO APL PORTS AVAILABLE*. The device will automatically be returned to VTAM after the message is displayed.

4. The CLEAR key in *RUNNING* mode no longer generates an attention signal to *APL*. The CLEAR key in *SELECT* mode will continue to generate the attention signal. Therefore, the screen may now be cleared in *RUNNING* mode without interrupting the function that is running.

5. The format of the *SELECT* mode status/option area has been modified to accomodate interfacing with the 328x printers through VTAM. The *SELECT* mode options now look like this:

```
-------------------------------------- SELECT                            PAGE nnnn
  PRINTER NAME: ?                          PAGING:ON          <<<<<      FROM nnnn
        STATUS: OFF                                           >>>>>      TO   nnnn
```

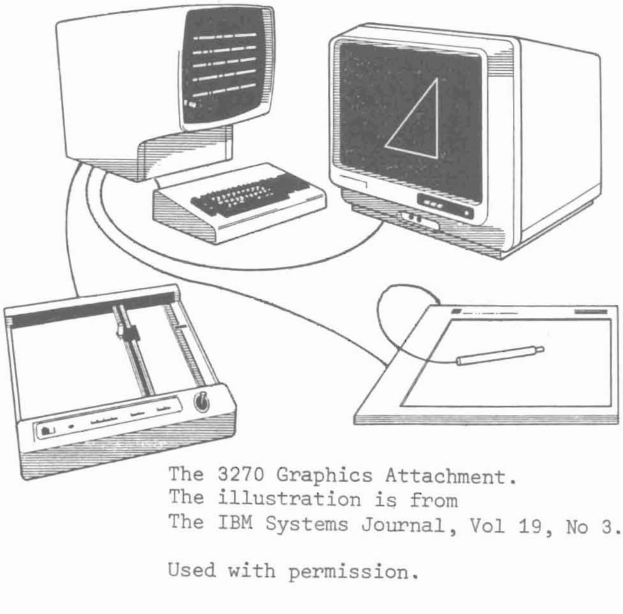
6. The printer name is the VTAM identification of the 328x printer that printer output will be directed to. If a VTAM printer name is associated with a 327x terminal during *APL* system generation, that name will appear initially in the *PRINTER NAME* field. If no printer name is provided, "*?*" will appear in this field. To change the printer associated with the terminal, modify this field and depress ENTER. Up to eight characters will be accepted and the entry will be padded on the right with blanks. If a printer is already attached to the terminal or has been requested and the printer name field is modified, the first printer connection will be terminated. A request to change the printer name is invalid if the terminal is not signed on to *APL*. The field is reset to the original value (either the associated name or "*?*") when the terminal is signed off from *APL*.

7. The printer *STATUS* field reflects the disposition of the printer. Initially the printer is *OFF*. The first detection of the field will cause a request to VTAM for the designated printer. The *STATUS* will then change to *ACQUIRING*, which will remain until a response from VTAM is received. Then, on the next status display following the response from VTAM, the status will be changed to *ON* if the request is satisfied, or *INVALID* if there is a problem (note that a status redisplay requires a user input). The problem could be an invalid name or a device that is unknown or unavailable to VTAM. Unless there is a problem, VTAM does not reply until the printer is available; therefore, the *IN USE* message is no longer used. While waiting for the printer, the terminal session can continue normally. Detection of the printer

STATUS keywords *ACQUIRING* or *ON* will terminate the connection or request and return the keyword to *OFF*. If VTAM terminates the printer connection during a session, the keyword *LOST* will appear once in the *SELECT* mode display following the termination. The printer connection is automatically terminated when the terminal is signed off *APL*. Detection of the printer *STATUS* field is invalid if the terminal is not signed on to *APL* or if there is no printer name provided.

8. In the *SELECT* mode it is now possible to enter some multiple requests. For example, the *PRINTER NAME* field can be modified and the *STATUS* field detected in one input. Or, the *PAGE* number field and the *PRINTER NAME* field can be modified in one input. Any combinations of requests that are not contradictory will be satisfied. If, however, the *PAGE* number field is modified and the keyword >>>>> detected, only the last request will be honoured.

The release date for all of these changes is still a little uncertain; there are hardware changes that we have to implement first. Ports will be converted as soon as possible. Some may be converted within a month; all should be converted by year-end.



The 3270 Graphics Attachment.
The illustration is from
The IBM Systems Journal, Vol 19, No 3.

Used with permission.

Coming soon!

3270 Full-screen support

Based on the many requests from *APL* users, we will soon be offering 3270 full-screen support, in the form of an auxiliary processor, AP124x. This processor isn't new, but it is new to *APLSV*.

The release date is still a little uncertain; there are hardware changes that we have to implement first. Ports will be converted as soon as possible. Some may be converted within a month. All should be converted by year-end.

A quick overview

Using AP124x is somewhat similar to using TSIO (AP370): two shared variables are required, a control variable and a data variable, whose names must respectively begin with *CTL* and *DAT*. Conceptually, you may think of the *DAT* variable as a variable that is shared between your *APL* function and the screen. When you specify a screen-load of data into the *DAT* variable, and specify "write" codes for *CTL*, the contents of the *DAT* variable replace the normal screen. In similar fashion, when the user types in anything, it replaces portions of the *DAT* variable and can be read by your *APL* function.

Normal screen operations will look the same as it always did... full screen management is only invoked by calling AP124x (using 124 ⎕*SVO* ---). The normal screen will return upon return to immediate execution, when a normal I/O request is issued, when the PA2 key is depressed, or when an *APL* error occurs.

In addition to full screen management, AP124x also allows control of the 3277 Graphics Attachment RPQ, with graphic output displayed on a Tektronix 618 display screen.

Please note that what we are offering is strictly a full-screen management auxiliary processor; in particular, it does not include a full-screen function editor. Such an editor is a totally separate issue, and is not being offered now.

To help to you get started using AP124x, we have developed a workspace with "building-block" aids:)*LOAD* 30 *FULLSCREEN*. This workspace contains various functions for screen management, for offering the shared variables, and for checking return codes.

CITING OUR SOURCES: No, alas, we did not write the following description; we obtained it with a five-finger discount. Since what we will be offering is AP124x (as opposed to AP124), one would assume that we would have stolen the description from the AP124x manual. Not quite true. We liked the write-up in the AP124 manual (personal preference), so we copied the AP124 text [from *VSAPL* for VSPC: Terminal User's Guide, SH20-9066], and then modified it to reflect AP124x [from *VSAPL* Extended Editor and Full Screen Manager, SH20-2341], and added local changes to reflect our *APLSV* environment. Since those manuals do not apply exactly to this system, we cannot recommend that you attempt to use them for *APLSV*.

Full Screen Management Auxiliary Processor

The Full Screen Management Auxiliary Processor allows you to control the screen format of an IBM 3270 display device through an *APL* defined function. In addition, it allows your application to:

- Write to the formatted screen
- Read from the formatted screen
- Erase screen fields
- Copy screen images to a printer
  [Not available in the initial release]
- Condition screen fields for light pen usage
- Read Program Function (PF) and Program Attention (PA) keys

Your *APL* application requests screen management services by assigning to the control variable a numeric scalar or vector that specifies the requested action. In response, the auxiliary processor issues a return code in the control variable indicating whether or not that was successful. If data is to be sent to or from the screen, it's transmitted in the data variable.

When a screen management service request is issued, the IBM 3270 enters full screen mode. When this happens, your function is in full control of the screen and may issue any additional screen management requests. When full screen mode is interrupted, the 3270 is returned to the default screen mode, and the most recent standard formatted screen is displayed.

Full screen mode is interrupted if the application returns to immediate execution mode for any reason; for example, if:

- an interrupt signal is issued (by pressing the PA2 key)
- a normal (non-screen management) input/output request is issued
- an error message is issued

When the 3270 leaves full screen mode and returns to normal mode, the current non-standard screen image is saved. It will be restored when the next full-screen management service request is issued.

Since *APL* issues a normal input/output request when a defined function completes execution, the duration of the full screen image may be extremely short. Your application must provide an intervening delay (such as a screen management read request) to extend the duration of the full screen image.

Screen Management - General Information

To use the Full Screen Management Auxiliary Processor, certain general information about the screen and its attributes is required. This information is provided in the discussions below.

Logical Screens

The Full Screen Management Auxiliary Processor is capable of handling multiple logical screens, each of which is defined to be exactly the size of the physical screen. Each logical screen is accessed through a pair of shared variables. When you share a pair of variables a logical screen is created, and upon retraction of the variables, the logical screen is destroyed. Each pair of variables is kept locically separate from the other variables, and the limit to the number of logical screens that may be open at any time is controlled by your shared variable quota and by system capacity limits.

Screen Fields

The Full Screen Management Auxiliary Processor logically views the screen of an IBM 3270 display device in terms of rectangular areas called screen fields. It is only these areas that data can be entered or displayed. Each screen field has a starting location, a width and height that your application defines when it formats the screen. The starting position of each field is the row and column address of the upper left-hand character in the field. [The upper left-hand position of the screen is row 1 column 1.]

Field Attributes

Each screen field has associated with it certain field attributes that qualify its content. For instance a field attribute may indicate that a field is to contain alphabetic or numeric data or that it is to be light pen sensitive. Your application can set the attributes for a field through various screen management service requests. If attributes are not explicitly set, the auxiliary processor supplies default values.

The 3270 Terminal System notes the attributes of a screen field by preceding the field with a column of attribute characters. These characters display as blanks and are not considered as part of the field. If a screen field begins in column 1, its attribute characters are wrapped around the screen, that is the characters will occupy column 80 in the previous row. A screen field that begins in row 1, column 1, will have an attribute character in column 80 on the bottom row of the screen.

It is generally good practice to leave at least one column for attribute characters between contiguous screen fiel'ds. Otherwise, the column of attribute characters preceding the right-hand field will obscure any data written to the last column of the left-hand field.

Light Pen Fields

Unlike other screen fields, light pen fields must contain a column of designator characters. The designator characters must be supplied by your application program... if they are not provided by your function, default values will be supplied by the auxiliary processor. The IBM 3270 Display System requires that the attribute characters and the designator characters must always be on the same row as the light pen field; therefore a light pen field may not be defined starting in column 1 because the display device hardware does not allow attribute characters to be wrapped around to the previous row for light pen fields. Fields adjacent to a light pen field on the same row must be separated by at least three additional blank columns. These blank columns may be either inside or outside of the data areas of the affected screen fields. A maximum of 12 light pen fields may precede the last light pen field on any given row. When mixing light pen and non-light pen fields, a maximum of 14 mixed fields may precede the last light pen field on any given row.

The designator column defines the two types of light pen fields: selection fields and attention fields. Each type of field performs a different light pen operation.

For a selection field, the designator column displays as question marks (?) or greater-than signs (>). When the light pen detects on a selection field, a designator character for the field is automatically changed on the screen from "?" to ">" or from ">" to "?" to provide visible indication to you that the detection is successful. If a mistake was made and you detect on the same field, the ">" reverts to a "?".

At the time an interrupt is generated by the user, the light pen selection fields that contain a ">" as the first character of one of their rows will be flagged as modified. Since the designator column is supplied by your function, certain fields may be "pre-selected" by setting the associated designator characters to a greater-than sign (>).

For an attention field, the designator column displays as blanks. A detection on an attention field completes the current input operation. The detected attention field and any previously detected selection fields are returned to your application.

It is important to remember that your function is responsible for supplying the the correct designator character for the field type (? or > for selection, blank for attention). If an incorrect character is supplied for a selection field, it will be changed to a "?". If an incorrect character is supplied for an attention field, it will be changed to a blank.

Using the Full Screen Management Auxiliary Processor

The following discussions describe the operations that can be requested through the Full Screen Management Auxiliary Processor.

- *CTL* represents the name of the control variable.
- *DAT* represents the name of the data variable.

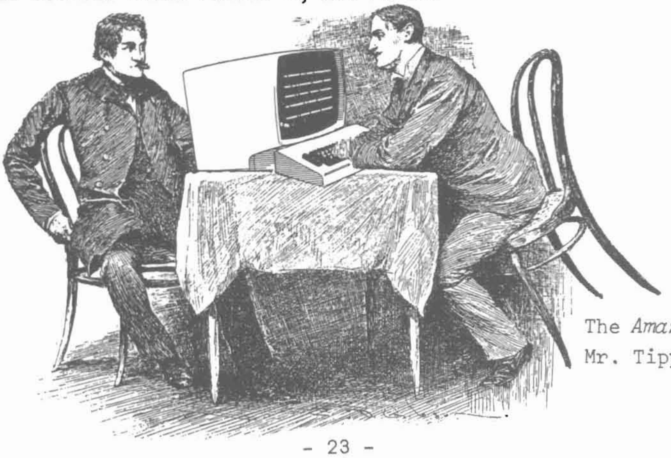
These items should be replaced with names that are approriate to your application when you issue your service requests. The names must begin with *CTL* and *DAT*, and must be no longer than 12 characters (total). The discussions assume that sharing has been completed for the control and data variables, and that the device is a 3270.

Formatting the Screen

This operation is used to describe the position and size of all screen fields. A request to format the display screen is made through the following assignments:

```
DAT←format
CTL←1
```

where format is a 4-, 5-, or 6-column numeric matrix with one row for each screen field to be formatted. If only one field is to be formatted, format can be a numeric vector, but is treated as a one-row matrix. The first (or only) row of the matrix defines the first screen field, the second row defines the format of the second screen field, and so on. The first screen field, unless redefined, becomes field number 1 for subsequent screen management operations; the second screen field becomes field number 2, and so on.



The *Amazing* Mr. Tippit!

The first four elements of <u>each row</u> of the matrix contain:

[1] <u>FIELD ROW</u>: the display screen row (origin 1) at which the field begins.

[2] <u>FIELD COLUMN</u>: the display screen column (origin 1) at which the field begins.

[3] <u>FIELD HEIGHT</u>: the height (number of rows) of the field.

[4] <u>FIELD WIDTH</u>: the width (number of columns) of the field.

The 5th and 6th elements of each row of the matrix are optional. If present, they contain:

[5] <u>FIELD TYPE</u>: the type of attribute of the field:

```
  0 - character input/output allowed
  1 - numeric character input/output allowed
* 2 - character output only    [default]
  3 - character output/light pen interruptable
  4 - character output/light pen selectable
```

[6] <u>FIELD INTENSITY</u>: the intensity attribute of the field:

```
  0 - off or don't display
* 1 - normal intensity    [default]
  2 - highlighted intensity
```

The starting position (upper left-hand corner) of a field must be a valid screen position. The enclosed area described by the starting position and the field height and width must not extend beyond the screen boundaries. (Except for light pen fields, fields with height 1 may wrap to succeeding rows.)

A zero in any row of the matrix effectively "undefines" a field. This removes the field from the formatted screen area but does not change the field numbers associated with the remaining fields.

When a display screen is formatted, the new screen format overlays the previous screen format. The new screen format is transmitted to the display screen on the next read or write operation.

Initially, the display screen contains only one field which covers the entire screen area.

Example One

The following illustrates a simple example of formatting. Here, a user defines two screen fields. the first field is to begin at row 3 column 5; it is to have a height of 10 rows and a width of 15 columns. The second field begins at row 3 column 30; it is to have a height of 12 rows and a width of 25 columns.

The user starts by defining the following formatting matrix "*MATX*", to be used as the argument to the function "*FORMAT*":

```
          MATX
      3  5 10 15
      3 30 12 25

     ∇ RET←P OFFER M;C;A
[1]   ⍝SHARED VAR OFFER TO PROCESSOR P
[2]    C←0
[3]   L0:→(2∨.=RET←P ⎕SVO M)/L1
[4]   ⍝DELAY AND TRY AGAIN
[5]    A←⎕DL 1
[6]    C←C+1
[7]    →(C≤5)/L0
[8]    ⎕←'PROCESSOR NOT AVAILABLE ... RETURN CODE = ',⍕RET
[9]    →0
[10]  ⍝SET CONTROL VECTOR FOR ASYNCHRONOUS CASE
[11]  L1:C← 1 0 1 0 ⎕SVC((¯2↑1,⍴M)⍴M)[⎕IO;]
     ∇

     ∇ RET←FORMAT MAT
[1]   ⍝DATA VARIABLE ASSIGNED FORMATTING MATRIX
[2]    DAT←MAT
[3]   ⍝ISSUE FORMAT REQUEST, ASSIGN RETURN CODE
[4]    CTL←1
[5]    RET←CTL
     ∇

     ∇ EXAMPLE1;CTL;DAT;RET
[1]   ⍝SHARE VARIABLES WITH FULL SCREEN MANAGER AND TEST
[2]    →(2≠1↑RET←124 OFFER 2 3 ⍴'CTLDAT')/0
[3]   ⍝FORMAT MATRIX AND TEST
[4]   L1:→(0=RET←FORMAT MATX)/L2
[5]    ⎕←'SCREEN NOT FORMATTED ... RETURN CODE = ',⍕RET
[6]    →0
[7]   L2:'... [CONTINUE PROCESSING]'
     ∇
```

Writing to the Screen and Erasing Screen Fields

This operation is used to write data to or erase data from one or more screen fields.

Writing

A request to write to the screen is made like this:

```
DAT←data              ...or...   DAT←data
CTL←2,fieldnum                   CTL←4,fieldnum

(2 ↔ Immediate write)          (4 ↔ Write to screen only
                               at next read-screen or
                               immediate write)
```

...where _data_ is a matrix of characters, each row of which contains the data for the corresponding field number in _fieldnum_, and _fieldnum_ is a numeric vector of one or more field numbers. Each field number represents a screen field to be written and corresponds to the respective row in the formatting operation matrix that defined that field.

When a write operation is executed, the auxiliary processor maps the data in the first row of _data_ into the screen field represented by the first element of _fieldnum_, the second row of _data_ into the screen field represented by the second element in _fieldnum_, and so on. If the height of a screen field format is greater than one, you must supply the data as a vector, not as a matrix. If several fields are being supplied at once, the data should be a matrix where each row is the ravel of the data for a given field (padded with blanks as necessary). The auxiliary processor automatically skips over any unformatted areas of the screen. If a field number in _fieldnum_ is larger than the number of rows specified in the format operation matrix, or if the field number does not currently apply to a defined field, that field number and the corresponding row of data are ignored. Similarly, if _data_ has more rows than field numbers in _fieldnum_, the extra rows are ignored.

Each line of a screen field is filled with data from left to right, starting at the beginning of the field. Any trailing blanks padding out a field line are replaced with nulls on the screen, subject to the setting of the field attributes. If too much or too little data is specified in a row of _data_, the data is respectively truncated, or extended with nulls or blanks to fit the full field area (we'll show how to choose whether nulls or blanks get used in just a bit).

Any data written to an attribute character position or to a designator character position will be obscured.

Data written will be displayed only momentarily. To maintain the display, a read request must follow the write request.

Erasing

The write operation also doubles as an erase operation, where each erased field is filled with null characters. You can erase data from a screen field in two ways:

1. ...in any write request, by listing the field number in _fieldnum_ without supply a corresponding row in _data_. If _data_ is empty, all fields identified in _fieldnum_ are erased. Fields not identified in _fieldnum_ are not affected by this operation.

2. ...in the first write operation after a format operation, by not including the field number in _fieldnum_. Any fields defined in the previous format operation and not specified in the write operation will be erased. (Subsequent write operations will not affect similarly unspecified fields.)

Example Two

As an example of writing and erasing, the function below formats four fields and then writes to them. *WRITE* uses two arguments. The left argument tells when the write is to occur (immediate or delayed), and the right argument is the data to be written.

```
      ρMATX                             ρFIELDDATA
  4 6                             4 12

      MATX                              FIELDDATA
    3  5  1 12  2  1              FIRST INPUT
    4  5  1 12  2  1              SECOND INPUT
    3 18  1  3  1  1              ___
    4 18  1  3  1  1              ___


     ∇ RET←WHEN WRITE FIELDS
  [1]  ⍝DATA VARIABLE ASSIGNED FIELDS
  [2]   DAT←FIELDS
  [3]  ⍝ISSUE WRITE COMMAND, ASSIGN RETURN CODE
  [4]   CTL←WHEN,⍳1↑ρFIELDS
  [5]   RET←CTL
     ∇

     ∇ EXAMPLE2;CTL;DAT;RET
  [1]  ⍝CONTINUATION OF EXAMPLE SHOWN IN 'EXAMPLE1'
  [2]   →(2≠1↑RET←124 OFFER 2 3 ρ'CTLDAT')/0
  [3]  L1:→(0=RET←FORMAT MATX)/L2
  [4]   ⎕←'SCREEN NOT FORMATTED ... RETURN CODE = ',⍕RET
  [5]   →0
  [6]  ⍝WRITE FIELDS TO SCREEN AND TEST
  [7]  ⍝ 2=IMMEDIATE WRITE  ...  4=DELAYED WRITE
  [8]  L2:→(0=RET←4 WRITE FIELDS)/L3
  [9]   ⎕←'SCREEN NOT WRITTEN ... RETURN CODE = ',⍕RET
  [10] →0
  [11] L3:'... [CONTINUE PROCESSING]'
     ∇
```

Reading from the screen

This operation is a two-step process which is used to read input from the screen. Although both steps are required to read screen fields into your workspace, the first step alone can be used to determine the nature of your input.

Step 1: Read and Wait

This step serves three purposes: first, it directs the auxiliary processor to wait for you to complete the current input operation; second, it tells it to return information about the current screen; third, based on how you completed input, it may tell it to read all the defined fields on the screen into an internal data area. Data is read only if you completed the input operation using the ENTER key or a PF key. Your function requests this operation by specifying:

```
       CTL←3
```

In response, the data variable will contain a vector of one or more numbers that indicate:

- ...the type of action that completed the current input operation. This is returned independently of how input was completed.
- ...the current cursor position. This information is returned if you completed the input operation using the ENTER key, a PF key or the light pen.
- ...the field numbers of the <u>modified</u> fields. A field is modified when you enter data into it and stays modified until your application writes to the screen. This information is returned if you completed the input operation using the ENTER key or a PF key.
- ...the field numbers of any light pen sensitive field selected (via the light pen) in the current input operation.

Step 2: Get the Data

This step completes the read operation. It is used to obtain one or more screen fields (usually the modified fields indicated in the read status vector) after step 1 has been performed. To request this step your function specifies:

```
       CTL←5,fieldnum
```

...where <u>fieldnum</u> is a numeric vector of one or more field numbers. Each field number represents a screen field to be read and corresponds to the respective row in the formatting operation matrix that defined that field.

When the operation is complete, the data variable contains a matrix of characters, each row of which is the data for the corresponding field in <u>fieldnum</u>. The matrix is padded to the right with blanks so that the number of columns is equal to the length of the longest field.

Note that the data returned in this step might be altered if your application issued any intervening write or format request or changed any field attributes.

DAT variable returned by read and test requests

User Action	Vector Elements 1	2	3	4	5	6...
	Completion Code	Modifier	Cursor Position Field[1]	Row[2]	Col[2]	Fields
Enter Key	0	0	fldnum	row	column	fieldnums[3]
PF Keys	1	1-24	fldnum	row	column	fieldnums[3]
Light Pen	2	0	fldnum	row	column	fieldnums[4]
Badge Reader	3	0-1[5]	fldnum	row	column	fieldnums[3]
PA Keys	4	1-3[6]	---	---	---	---
Clear Key	5[7]	---	---	---	---	---
No Input	6	---	---	---	---	---

NOTES:

[1] Indicates the field number of the field containing the cursor when input was completed. If this element is 0, the cursor was not found in a defined field. Elements 4 and 5 are then physical position indicators (relative to row 1, column 1).

[2] Indicates the position of the cursor when input was completed. The position is indicated by the row and column relative to the first row and column position in the field.

[3] Indicates one field number for each field modified since the last preceding write operation. If no fields were modified, this element is not returned.

[4] Indicates one field number of each selected light pen sensitive field.

[5] Indicates 0 for an accepted badge, 1 for a rejected badge.

[6] Pressing the PA2 key generates a weak attention signal. The function is suspended, and consequently the screen is returned to normal screen mode.

[7] If the Clear Key was used, the current screen format and field attributes are re-established at the next read or write operation.

Example Three

The following function is an example of a complete format, write, and read session. The function *READ* initiates the read, waits for the user to complete the input. The results are stored in two global variables, *INFO* and *DATA*. *INFO* contains the user's action and the modified fields; *DATA* is assigned the values contained in the fields witch were read. An application can later reference these variables as needed.

(*MATX* and *FIELDDATA* are found in Example Two)

```
    ∇ RET←READ FIELDS
[1]  ⍝ISSUE READ AND WAIT COMMAND, CHECK RETURN CODE
[2]   CTL←3
[3]   →(0≠RET←CTL)/0
[4]  ⍝ASSIGN USER ACTION TO GLOBAL VARIABLE INFO
[5]   INFO←DAT
[6]  ⍝ISSUE GET DATA COMMAND, CHECK RETURN CODE
[7]   CTL←5,FIELDS
[8]   →(0≠RET←CTL)/0
[9]  ⍝ASSIGN DATA TO GLOBAL VARIABLE DATA
[10]  DATA←DAT
    ∇

    ∇ EXAMPLE3;CTL;DAT;RET
[1]  ⍝CONTINUATION OF EXAMPLE SHOWN IN 'EXAMPLE2'
[2]   →(2≠1↑RET←124 OFFER 2 3 ⍴'CTLDAT')/0
[3]  L1:→(0=RET←FORMAT MATX)/L2
[4]   ⎕←'SCREEN NOT FORMATTED ... RETURN CODE = ',⍕RET
[5]   →0
[6]  L2:→(0=RET←4 WRITE FIELDS)/L3
[7]   ⎕←'SCREEN NOT WRITTEN ... RETURN CODE = ',⍕RET
[8]   →0
[9]  ⍝READ FILEDS 3 4 AND TEST
[10] L3:→(0=RET←READ 3 4)/L4
[11]  ⎕←'SCREEN NOT READ ... RETURN CODE = ',⍕RET
[12]  →0
[13] L4:'... [CONTINUE PROCESSING]'
    ∇
```

Modifying Field Attributes

These operations are used to explicitly set the attributes of one or more screen fields. Three kinds of attributes can be set:

- the <u>type of data</u> permitted in the field
- the <u>light pen status</u> of the field
- the <u>display intensity</u> of the field

In the descriptions of each of these cases the term <u>fieldnum</u> is a numeric vector of one or more field numbers. Each field number represents a screen field whose attributes are to be set and corresponds to the respective row in the format operation matrix which defined that field.

Modifying Data Type and Light Pen Status

This operation is used to indicate what type of data is permitted in a field and whether or not the field is light pen sensitive. It takes effect at the next screen management read or write operation and applies until changed explicitly or until the screen is reformatted. Your application can request this operation by specifying:

```
DAT←type
CTL←6,fieldnum
```

...where type is a numeric vector of field type indicators. Each value in type indicates the data type and light pen status for the corresponding field in fieldnum. If type is a single value, it is the field type indicator for all the fields specified in fieldnum.

The field type indicators are:

```
  0 - alphabetic or numeric input/output allowed
  1 - numeric input/output allowed
* 2 - alphabetic or numeric output only     [default]
  3 - alphabetic or numeric output allowed/light pen
      interruptable
  4 - alphabetic or numeric output allowed/light pen
      selectable
```

As an example, the following two assignments set the data types and light pen status of two previously formatted screen fields (screen fields 1 and 2). Both fields are set to accept alphabetic or numeric output; the second field in addition, is made light pen interruptable. Sharing for both the control and data variables are assumed to be complete:

```
DAT←2 3
CTL←6 1 2
```

Modifying Field Intensity

This operation is used to set the display intensity of one or more fields. It takes effect at the next screen management read or write operation and applies until changed explicitly or until the screen is reformatted. Your application can request this operation by specifying:

```
DAT←intensity
CTL←7,fieldnum
```

...where intensity is a numeric vector of intensity indicators. Each value in intensity indicates the display intensity of the corresponding fields in fieldnum. If intensity is a single value, it is the field type indicator for all the fields specified in fieldnum.

The intensity indicators are:

```
      0 - off or don't display
    * 1 - normal intensity    [default]
      2 - highlighted intensity
```

As an example, the following two assignments set the display intensity of a previously formatted screen field (screen field 2) to be highlighted. Sharing for the control and data variables are assumed to be complete:

```
      DAT←2
      CTL←7 2
```

Modifying Input Field Attributes

This operation is used to modify additional attributes for one or more fields. It takes effect at the next screen management read or write operation and applies until changed explicitly or until the screen is reformatted. The additional attributes that may be specified control the handling of (1) trailing nulls, and (2) an "Auto-skip" feature. These attributes are only meaningful for input fields, but may be specified for any field.

Your application can request this operation by specifying:

```
      DAT←attribute
      CTL←16,fieldnum
```

...where attribute is a numeric vector of attribute indicators. Each value in attribute indicates the attribute of the corresponding field in fieldnum. If attribute is a single value, it is the attribute indicator for all of the fields specified in fieldnum.

The attribute indicators are:

```
      0 - Non-Autoskip, no trailing blank processing
      1 -     Autoskip, no trailing blank processing
      2 - Non-Autoskip,    trailing blank processing
    * 3 -     Autoskip,    trailing blank processing  [default]
```

If "Auto-skip" is specified, at the end of the field the attribute byte is set to automatically skip to the next input field. This attribute is meaningful only for input fields.

If trailing blank processing is specified, all trailing blanks in the user's data are converted to nulls upon presentation to the physical screen. This allows the terminal user to use the INSERT MODE key on the 3270 to insert data into the field. If this option is turned off, the user's data is not modified upon presentation to the 3270 (i.e., trailing blanks will remain true blanks on the screen).

Reading the Screen Format

The Full Screen Management Auxiliary Processor provides an easy way of determining what format matrix it is currently using. A request to display the current format matrix is made like this:

```
      CTL←9
      format←DAT
```

...where <u>format</u> contains one row for each field, up to the highest valid field defined, and is six columns wide. If a new format is pending, the new format matrix is returned.

Printing a Screen Image

During its interaction with the Full Screen Management Auxiliary Processor, your function may direct a copy of the current screen image to the printer by specifying:

```
      CTL←10              [Not available in the initial release]
```

The printer must have been previously defined. Refer to the previous article.

Sounding the Alarm

In your screen operations, you may find it useful at times to sound an audible alarm. For instance, you might want an alarm sounded when a field of particular importance is filled. Your application can request the auxiliary processor to sound the alarm through the assignment:

```
      CTL←11
```

This request takes effect at the next screen management read or write operation.

Setting the Cursor

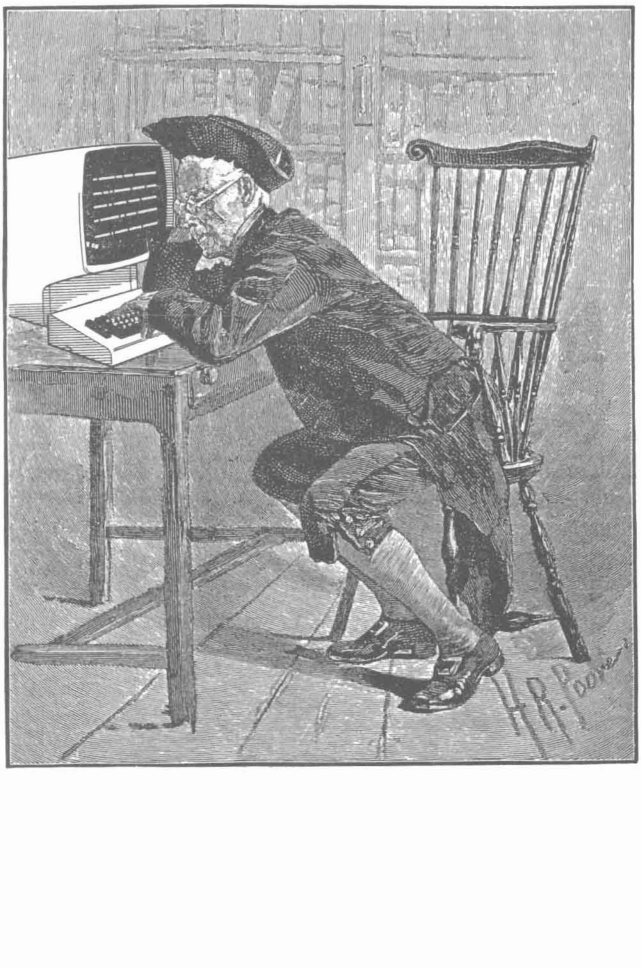
This operation is used to position the cursor at the end of the next read, write, or erase operation. Your application can request this operation by specifying:

```
      DAT←fieldnum,row,col
      CTL←12
```

If the first element is zero, the second and third elements are interpreted as the row and column position of the cursor from the upper left-hand position on the screen.

If either of the last two elements is zero or negative, the position of the cursor will not be changed in the next read, write, or erase operation. [This too, is the default condition until you issue this service request.]

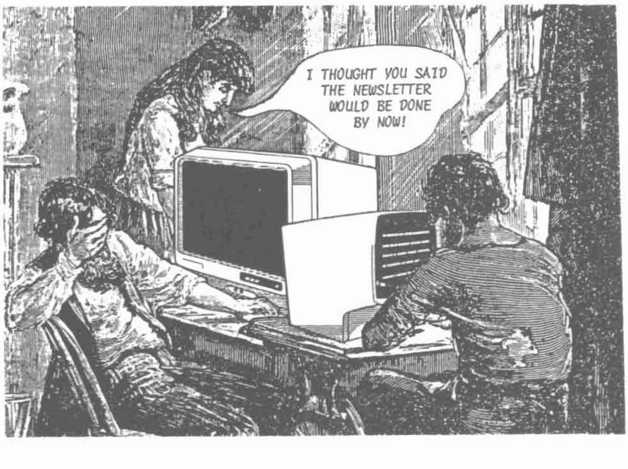
When the auxiliary processor is first invoked, the cursor appears on the screen in position 1, 1. The cursor will default to this position after any new screen format takes effect.



AP124x Return Codes

The return codes from the Full Screen Management Auxiliary Processor (AP124x) are all scalar integers.

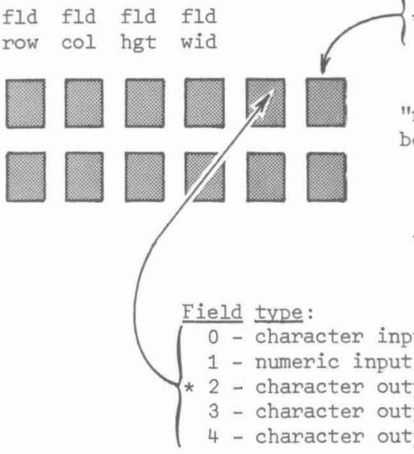
Return Code	Description
0	Successful
1	*CTL* nonce error (command not currently implemented)
11	*CTL* rank error
12	*CTL* length error
13	*CTL* domain error
14	Invalid command
15	Request to position cursor in an undefined field
21	*DAT* rank error
22	*DAT* length error
23	*DAT* domain error
24	*DAT* variable not shared
30	Invalid field number
32	Defined field extends beyond screen
33	Reference outside field definition
35	Light pen field starts in column 1
36	Light pen field (with height 1) not contained in one physical screen line
37	Invalid field type
38	Invalid field intensity
39	"GET" or "PUT" for an undefined field
41	*DAT* not specified in correct sequence
42	*DAT* not referenced in correct sequence
54	Printer is not available
91	Physical field table overflow
92	Physical field table error, interrupt field not found
95	I/O error



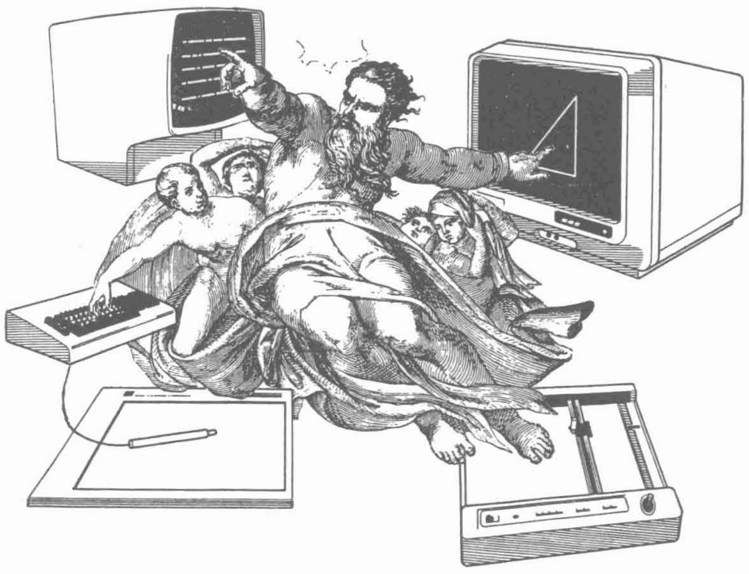
CTL←	("fieldnum" identifies the field number of format matrix, shown on the next page... "fieldnum" is always specified in origin 1)
0	Delayed clear of screen (see also 20)
1	Format (see description of *DAT* on next page)
1,fieldnum	Reformat (see description of *DAT* on next page)
2,fieldnum	Immediate write to screen (see also 4)
3	Read and wait
4,fieldnum	Delayed write to screen (see also 2)
5,fieldnum	Get data
6,fieldnum	Modify field type ...set *DAT* to field type: 0 - alphabetic or numeric input/output allowed 1 - numeric input/output allowed * 2 - alphabetic or numeric output only [default] 3 - alphabetic or numeric output allowed/light pen interruptable 4 - alphabetic or numeric output allowed/light pen selectable
7,fieldnum	Modify field intensity ...set *DAT* to intensity: 0 - off or don't display * 1 - normal intensity [default] 2 - highlighted intensity
9	Read screen format
10	Copy screen image to printer
11	Sound alarm
12	Position the cursor ...set *DAT* to three-element vector: number of screen field, fieldnum, column [If 1st element is 0, 2nd and 3rd are position from top left corner of screen]
16,fieldnum	Modify field attribute ...set *DAT* to attribute: 0 - Non-Autoskip, no trailing blank processing 1 - Autoskip, no trailing blank processing 2 - Non-Autoskip, trailing blank processing * 3 - Autoskip, trailing blank processing [default]
20	Immediate clear of screen (see also 0)

* ↔ Default value

```
Format:  (CTL←1)

DAT←      (DAT may be "n" rows by 4, 5, or 6 columns; if DAT
           is a vector, it is treated as a 1-row matrix)

←--- Origin 1 ---→ ←Optional→      Screen intensity:
                                      0 - off or don't display
fld  fld  fld  fld                  * 1 - normal intensity
row  col  hgt  wid                    2 - highlighted
```



```
"n"-rows allowed, rows will
be referred to by "fieldnum"

* ↔ Default value

Field type:
  0 - character input/output allowed
  1 - numeric input/output allowed
* 2 - character output only
  3 - character output/light pen interruptable
  4 - character output/light pen selectable
```



Northumbrian *APL*er, apprehended *en flagrante delicto*
with an underground newsletter operation

Wednesday, 697 AD

New APL character available: Dollar-sign

A new character has been added to *APLSV*: the dollar-sign is now available for use as a decorator on reports. It appears in the atomic vector as ⎕*AV*[210] (in origin 1).

Note that the dollar-sign is neither a function nor an operator; it will always produce an error when used outside of quotes, but it is recognized by the system for use as a decoration symbol. You may find it to be a particularly useful companion to the new Picture Format primitive.

To enter the character from a 3277, you must first go to "*APL* OFF"-mode. Since the dollar-sign wasn't an *APL* character back when the *APL* feature was designed for the 3277, it isn't available in "*APL* ON"-mode. Please realize that any other characters that you enter in that manner are <u>not</u> *APL* characters, and may cause problems [e.g., they will be rejected by ⎕*FX*, and they won't display on other terminals].

On typewriter-type terminals (2741, MCST, etc), the dollar-sign may be entered by overstriking "*S*" and "/".

On a 5100 or 5110 (in communications mode) the dollar-sign key is not supported, and the "*S*" and "/" overstrike will display as a blot symbol. As with the 3277, this is because the dollar-sign was not supported when those devices were designed. In normal entry mode, however, the character can be entered as a compound character (in "EDIT" mode, it can't be entered).

The translations for TSIO's *CODE=E* have also been extended to include the dollar-sign as a valid *APL* character.

Picture Format (a.k.a. "Format by Example")

One of the most common requirements of business data processing is formatting data for reports. This has sometimes been a difficult task, with the output often lacking the decorators that were desired for a truly readable report. How many of you, for example, have taken the trouble to insert commas into large numbers for readability? "454217329" becomes much more understandable as "454,217,329", but as nice as that might be on the final report, formatting it that way used to be a formidable task. But now, with Picture Format, such tasks become trivial.

The Picture Format primitive shares the same symbol with the other formatting primitives: "⍕". But while dyadic format has always used a <u>numeric</u> left argument in the past (5 0⍕*M*), it may now also use a character left argument (' 55,555.00'⍕*M*). When the left argument is a character string, the function is Picture Format.

There are, of course, very specific rules for the format of the left argument... but we'll come back to that. For now, let's just say that the left argument (or "pattern") shows *APL* a sample "picture" of what we want the results to look like.

Picture Format provides an <u>easy</u> method for you to:

⍕ Print numeric output with controlled commas: 16,777,215

⍕ Use any "negative" indicator that you wish, in case the *APL* "¯" symbol isn't available with the printer or typeball that you want to print your report on; use "-", "credit", or whatever you want

⍕ Optionally suppress fields that represent values of 0, so that they print as blank fields

⍕ Print values with leading or trailing zeros

⍕ Float a decorator, such as a dollar-sign, in against your data

⍕ Print numeric values in European notation, with a comma separating the integer and decimal portions of a number: 12,34

⍕ Display negative numbers within parentheses, as on accounting reports: (12.50)

...or, well, you name it.

Picture Format often can make short work of what had previously been complex formatting jobs.

Using Picture Format, we can do a lot of formatting with a minimal amount of programming. Let's say that we want to build a report function that will dress up the output for us. Here's a sample function that could do the formatting for us. Again, don't be concerned right now with just what the rules are for coding up the left argument for Picture Format... we'll get to that in a bit. For right now, simply notice how compact the actual formatting is when we use Picture Format:

```
      ∇ Z←REPORT M
  [1]  ⍝FORMAT ALL THE DATA; ADD DOLLAR SIGNS AND 'CR' NOTES; ADD TOTAL LINE:
  [2]   Z←'    $35,555.19_CR'⍕M,[1]+⌿M

  [3]  ⍝ADD COLUMN HEADINGS:
  [4]   Z←'        ITEM 1          ITEM 2          ITEM 3          ITEM 4    ',[1] Z

  [5]  ⍝ADD ROW MARKINGS:
  [6]   Z←(5 7 ⍴'GROUP  DEPT A:DEPT B:DEPT C:TOTAL: '),Z

  [7]  ⍝PUT BLANK LINES BETWEEN HEADING AND BODY, AND BETWEEN BODY AND TOTAL:
  [8]   Z← 1 0 1 1 1 0 1 \Z
      ∇
```

Here's the data that we're working with:

```
        ⍴DATA
  3 4
        DATA
  12345.67          1           ¯50.55          127.23
     34.15   ¯1234.56          4500            ¯222.5
    227.5           0        ¯56789              19.56
```

Running the "*REPORT*" function, we get this finished report:

```
        REPORT DATA
  GROUP        ITEM 1          ITEM 2          ITEM 3          ITEM 4

  DEPT A:  $12,345.67           $1.00          $50.55 CR    $127.23
  DEPT B:      $34.15   $1,234.56 CR    $4,500.00          $222.50 CR
  DEPT C:     $227.50                  $56,789.00 CR        $19.56

  TOTAL:   $12,607.32   $1,233.56 CR  $52,339.55 CR        $75.71 CR
```

* * *

One of the problems that you may have experienced in the past is specifying the proper numeric left argument for the format primitive such that the output properly lines up with your column headings. With Picture Format, the length of the left argument is the length of the result, so this becomes a good deal simpler:

```
    ∇ FMT V
[1]  ⍝PROBLEM:  WHEN USING FORMAT (⍕), IT'S DIFFICULT TO ALIGN COLUMNS:
[2]   ⎕←'OLD     USERS  CONNECT  COMPUTE  WORKSPACES   BLOCKS   TRACKS'
[3]   ⎕←'FMT:', 8 0 9 0 9 1 12 0 9 0 9 0 ⍕V
[4]   ⎕←''

[5]  ⍝SOLUTION:  PICTURE FORMAT ALLOWS HEADINGS AND FORMAT CONTROL TO ALIGN:
[6]   ⎕←'NEW     USERS  CONNECT  COMPUTE  WORKSPACES   BLOCKS   TRACKS'
[7]   ⎕←'FMT:  55,555   55,555    555.0      555,555  555,555  555,555'⍕V
[8]   ⎕←''
    ∇


      FMT V
OLD    USERS  CONNECT  COMPUTE  WORKSPACES   BLOCKS   TRACKS
FMT:    3513    11173    107.3       33658   306781   153390

NEW    USERS  CONNECT  COMPUTE  WORKSPACES   BLOCKS   TRACKS
FMT:   3,513   11,173    107.3      33,658  306,781  153,390
```

As we mentioned, the length of the result from Picture Format is the same as the length of its left argument (except for the case where Picture Format contains just one field, which will then apply to each column of data). Likewise, the positions of such things as commas and decimal points will match the position of these items in the output.

Fields within the left argument are typically separated by blanks (although we'll see a way to let other characters separate the fields, too). The number of fields in the left argument must match the last dimension of the data being formatted, although if there is only a single field, that's acceptable too... it will be used for every column of data. So then, what's a field? A field is a sequence of characters bounded by blanks (or the end of the pattern) containing at least one digit. If it doesn't have any digits, it's a decoration. That's allowed, too; this example has two fields:

```
          '| DATA 55 55 |'⍕2 2⍴10 20 30 40
      | DATA 10 20 |
      | DATA 30 40 |
```

The vertical bars and the word "*DATA*" aren't fields, because they don't contain any digits; they therefore become simple decorators.

Here's an application of the previous example, from workspace "1 *BILLING*":

```
    ∇ Z←RATEFMT V
[1]   Z←'RATES FOR 5555:   $ 5.50/CONNECT HOUR, $ 55.50/CPU MINUTE, $ 0.50/TRACK/WEEK'⍕V
    ∇
      RATEFMT 1980 4.25 21 .02
RATES FOR 1980:   $ 4.25/CONNECT HOUR, $ 21.00/CPU MINUTE, $ 0.02/TRACK/WEEK
```

Consider what its predecessor had to look like:

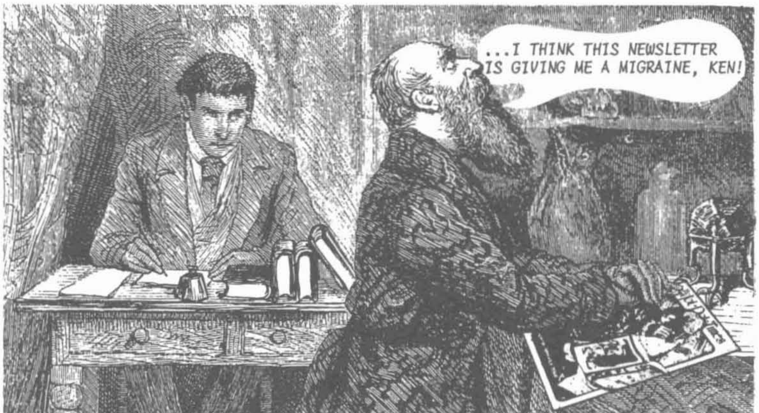
```
    ∇ Z←RATEFMT1 V
[1]   Z←'RATES FOR ',(⍕V[1]),':   $ ',(4 2 ⍕V⌈2]),'/CONNECT HOUR, $ ',(5 2 ⍕V[3]),'/CPU
      MINUTE, $ ',(4 2 ⍕V[4]),'/TRACK/WEEK'
    ∇
      RATEFMT1 1980 4.25 21 .02
RATES FOR 1980:   $ 4.25/CONNECT HOUR, $ 21.00/CPU MINUTE, $  .02/TRACK/WEEK
```

Digits in the pattern serve a dual purpose: they show where digits may appear in the result, but further, specific digits in the left argument have specific meanings regarding the formatting that is to take place. They are called "distinguished digits", and a table of them follows shortly.

Non-numeric characters in the left argument can be:

- simple decorators (like the example just shown),
- controlled decorators (such as a comma, which appears or is suppressed according to established conventions), or
- floating decorators (such as a dollar-sign, which can be made to nestle in against the left side of the data). The action of these floating decorators is also controlled by selecting which of the "distinguished digits" you use.

Further control is provided through the use of a new system variable, ⎕*FC* (Format Control). This variable acts as another (implicit) argument for Picture Format. A table explaining its operation is also following.



"Distinguished Digits" for Picture Format

[In general, use "5"'s for the pattern except where special handling is desired, as noted in the table]

1

Float the decorator in against the number only if the value is negative. [See notes below]

```
      ' -551.50'⍕ ¯1 10 ¯100
   -1.00   10.00 -100.00

      ' 551.50-'⍕ ¯1 10 ¯100
   1.00-  10.00  100.00-

      ' (55,551)'⍕ ¯10000 ¯1 10 ¯100
 (10,000)      (1)      10     (100)
```

NOTE that it is up to the applications programmer to provide a "negative" indicator that is appropriate to his own application. Picture Format provides the capability of using any sign that you wish ("¯", "-", "*CR*", etc.). If this is not done, e.g., if the pattern doesn't include "1"s or "2"s, a *DOMAIN ERROR* will result (see ⎕*FC*[4] in the "Description of ⎕*FC*" to over-ride this).

2

Float the decorator in against the number only if the value is non-negative. [See note below]

```
      ' +552.50'⍕ ¯1 10 ¯100
    1.00  +10.00  100.00
```

3

Float the decorator in against the number for all values (positive or negative). [See note below]

```
      ' $553.50'⍕ 1 10 100
   $1.00  $10.00 $100.00

      ' $553.10-'⍕ ¯1 10 ¯100
   $1.00-  $10.00  $100.00-

      ' $553.10CR'⍕ ¯1 10 ¯100
   $1.00CR  $10.00    $100.00CR
```

If only one of distinguished digits 1, 2, or 3 appears within a given pattern, its effect applies to both right and left floating decorations. If more than one appears, each one affects its own respective side.

4

Counteracts the action of a 1, 2, or 3, preventing it from affecting the other side of the field, which is then treated as a simple decorator.

```
      ' -551.40CR'⍕ ¯1 10 ¯100
    -1.00CR    10.00CR -100.00CR
```

5	Perform normal formatting, observing normal *APL* rules of removing leading and trailing zeros, except that a value of zero will display as all-blank. <u>Careful</u>, though, it's up to you to include an appropriate sign character if you expect any negative values, and a "1" or "2" to control it; see the "NOTE" on the previous page.
	
	`      '  555.55'⍕1.1 10.01 100 0 100.10`
	`    1.1    10.01  100             100.1`

6	The decorator to the right also marks the end of this field; treat it as though there's a blank between the fields, but also print the decorator.
	
	`      '05/55/55'⍕32580`
	`03/25/80`
	
	`      '06/06/05'⍕ 3 25 80`
	`03/25/80`

7	The next non-numeric character to the right is the symbol to be used for exponential notation ("*E*-format").
	
	`      '   ¯1.70E¯01'⍕ ¯.001 100 10000 ¯10000000000`
	`   ¯1.00E¯03     1.00E 02     1.00E 04    ¯1.00E 10`
	
	`      '   ¯1.70*¯10'⍕ ¯.001 100 10000 ¯10000000000`
	`   ¯1.00* ¯3     1.00*  2     1.00*  4    ¯1.00* 10`

8	"Check-protection": fill empty portions of the field with whatever character is in ⎕*FC*[3] (in origin 1). The default character is *.
	
	`      ' 8555.50'⍕ 1 10 100`
	` ***1.00 **10.00 *100.00`
	
	`      ' 5855.50'⍕ 1 10 100`
	`  **1.00  *10.00  100.00`

9	Pad with leading or trailing zeros out to this point (before or after the decimal point, respectively), unless the value is all zero (then use all blanks). [Compare with 0]
	
	`      '  555.59'⍕ 1.1 10.01 0 100`
	`    1.10   10.01           100.00`
	
	`      '  555.50'⍕ 1.1 10.01 0 100`
	`    1.10   10.01     .00   100.00`

0	Pad with leading or trailing zeros out to this point (before or after the decimal point, respectively). [Compare with 9]
	
	`      '  055.50'⍕ 1 10 100`
	`  001.00  010.00  100.00`

Description of ⎕*FC* -- Format Control

Default setting: ⎕*FC*←'.,*0_'

⎕*FC*[1]		Decimal point: the character that's to be substituted for the period where a decimal point is required in the result. This also affects numeric-left-argument format.
		
		`      ⎕FC[1]←'.'` `      ⎕FC[1]←','`
		`      '55.55'⍕12.34` → `12.34` ; `      '55.55'⍕12.34` → `12,34`
		`      5 2⍕12.34` → `12.34` ; `      5 2⍕12.34` → `12,34`

⎕*FC*[2]	,	Comma: the character to be substituted for the comma where a controlled comma is required in the result.
		
		`      ⎕FC[2]←','` `      ⎕FC[2]←' '`
		`      '5,555'⍕1234` → `1,234` ; `      '5,555'⍕1234` → `1 234`

⎕*FC*[3]	*	Check Protection character: the character to be printed in response to the "8"'s in the pattern.
		
		`      ⎕FC[3]←'*'` `      ⎕FC[3]←'/'`
		`      ' 855'⍕1 10 100` → ` **1 *10 100` ; `      ' 855'⍕1 10 100` → ` //1 /10 100`

⎕*FC*[4]	0	Overflow control: If the default character appears here, a value which is too large to fit into a field specified will cause a *DOMAIN ERROR*. If any other character appears here, the error will not occur and instead the offending field will be filled with the character specified. This also affects numeric-left-argument format.
		
		`      ⎕FC[4]←'0'` `      ⎕FC[4]←'?'`
		`      ' 00'⍕1 10 100` → *DOMAIN ERROR* ; `      ' 00'⍕1 10 100` → `01 10 ??`
		`      ' 00'⍕ 1 10 100` / `          ∧` ; `      3 0⍕10000` → `???`

⎕*FC*[5]	_	"Print-as-blank" character: any place that this character appears in the pattern, it will print as a blank; it functions normally in the analysis of the pattern, but is replaced by a blank in the result. In ⎕*FC*, it may not be a blank, period, comma, or a digit.
		`      ' $_35,555'⍕12345 250 5000`
		`$ 12,345   $ 250  $ 5,000`
		`      ' 15,555_CR'⍕¯12345 250 ¯5000`
		`12,345 CR    250     5,000 CR`

While ⎕*FC* currently contains five characters, it is recommended that you don't consider its length to be fixed. Future extensions could add additional elements.

The only valid current configuration for ⎕*FC* is five character elements. Any setting other than this will cause a ⎕*FC IMPLICIT ERROR* to be evoked when any use of dyadic format is attempted.

Note that the first two elements show what characters are to be printed where the decimal point and controlled comma are required in the result. The pattern is always coded using U.S. conventions; ⎕*FC* can be changed to allow display of British or other standards. This requirement for the pattern was done to allow an easy transfer of programmes between countries... a change to all the patterns in the workspace isn't needed for such a move, only a simple change to ⎕*FC*.

YES, SIRE, I KNOW WE OVERSHOT THE NEW WORLD ON THAT LAST TRIP, BUT WITH THIS NEW DYADIC EXECUTE, WE COULD HAVE CAUGHT THAT....
Special Reference Issue

New primitive function: Dyadic Execute

The execute function is now available in a new flavour. Previous discussions of execute have often alluded to the idea that ⍎⍞ can be used as "an alternative to ⎕ input offering more program control". Well, maybe, but any of you who have tried to actually do this have probably discovered that the problems start when the first user of your application types in an entry that's not a "well-formed *APL* expression":

```
      ⍎⍞
2+
⍎ SYNTAX ERROR
      2+
       ∧
```

A function that is trying to prompt for a character string that represents a vector of floating-point numbers, and then execute it to get the string into numeric form, may well spend most of its time simply ensuring that the execution is going to work; every possibility of an erroneous input must first be checked. Perhaps the classic example of this is using ⌹ to invert a matrix: there are rules governing the acceptability of the matrix for inversion, but checking the matrix will probably take longer than the inversion. A nice approach would be to simply try it, and back off if it fails. In the past this wasn't possible, since an error would halt the function. But now, with dyadic execute, this may easily be done.

Consider the case of ⍎*R*. If *R* can't be executed, an error message will be returned (e.g., ⍎ *SYNTAX ERROR*, ⍎ *LENGTH ERROR*, ⍎ *WS FULL*, etc.). The dyadic form, *L*⍎*R*, will return exactly the same result as ⍎*R* if the execution is successful; the left argument will be ignored. But if *R* can't be executed, the expression will be treated just as if it was ⍎*L*. In particular, if the left and right arguments are both invalid, an error message will be reported that will look just as if the expression had been ⍎*L*:

```
      ⍎ '3+'
⍎ SYNTAX ERROR
      3+
       ∧

      '2+2' ⍎ '3+'
4

      '2+' ⍎ '3+'
⍎ SYNTAX ERROR
      2+
       ∧
```

A particularly useful application of dyadic execute is '→*L*9'⍎*FOO*, in which any problem in the character string *FOO* which would prevent it from being executed will cause a branch to label *L*9.

Dyadic execute will switch arguments after any occurrence of an error in the right argument, regardless of the depth of the function calls that may have occurred in the right argument. For example, consider '→*L*9'⍎'*FN*', where *FN* is a function. If *FN* calls another function, *FN*2, which subsequently encounters a *DOMAIN ERROR*, the error will not be reported, but rather, the execute function will immediately abandon execution of the right argument, and instead will execute the left argument (→*L*9).

Be aware of a frequent trap: a common approach is to enter an expression such as *Z*←'→*L*9'⍎⍞ with the idea that an error would cause the function to branch. ...'taint so, McGee. If the input is executable, the expression can be viewed as *Z*←⍎⍞. But if it's not executable, the expression becomes *Z*←⍎'→*L*9', or *Z*←→*L*9 ...an immediate *SYNTAX ERROR*. Therefore, although it's longer, a bit slower, and somewhat more cumbersome, what's <u>really</u> needed is '→*L*9'⍎'*Z*←⍎⍞'.

* * *

Please realize that this primitive is <u>not</u> meant to be an all-encompassing coverage of generalized error side-tracking. There are many situations where recovering from an error during execution will not be possible. One thing that is admittedly absent is a manner of determining where the error occurred, and what type of error <u>would</u> have been reported (*LENGTH ERROR*, *DOMAIN ERROR*, etc.). But for situations in which you can anticipate a specific problem, and have a remedy for it, dyadic execute may be just the ticket.

* * *

Here is an example of a simple input checking function which will prompt for numeric data, and will re-prompt if the input can't be executed:

```
     ∇ Z←NUM T
[1]  ⍝03/12/80 → 09/06/80  MCGREW, 63C, KINGSTON
[2]  ⍝PROMPTS USER WITH MSG IN RT ARG, EXECUTES INPUT
[3]  L1:⍞←T
[4]   Z←⍞
[5]   →(Z∧.=' ')/L8
[6]   '→L9'⍎'Z←,⍎Z'
[7]   →0
[8]  L8:Z←⍳0
[9]   →0
[10] L9:⎕←'INVALID; PLEASE RE-TRY...'
[11]  →L1
     ∇
```

```
      R←NUM 'ENTER NUMERIC STRING:  '
ENTER NUMERIC STRING:  1 2 3 4.5.6
INVALID; PLEASE RE-TRY...
ENTER NUMERIC STRING:  ¯3.7¯
INVALID; PLEASE RE-TRY...
ENTER NUMERIC STRING:  1 2 3 4. 5.6
      ρR
5
      R
1 2 3 4 5.6

      R←NUM 'ENTER NUMERIC STRING:  '
ENTER NUMERIC STRING: ←—— [user presses "return"]
      ρR
0
```

Note that if the function were "simplified" a bit, it could become difficult for a well-meaning user to exit the function:

```
    ∇ Z←NUM2 T
[1]  ⍝03/12/80 → 09/07/80  MCGREW, 63C, KINGSTON
[2]  ⍝PROMPTS USER WITH MSG IN RT ARG, EXECUTES INPUT
[3]  L1:⍞←T
[4]   '→L9'⍎'Z←,⍎⍞'
[5]   →0
[6]  L9:⎕←'INVALID; PLEASE RE-TRY...'
[7]   →L1
    ∇
                          [The user calls the function,
                           but then decides to cancel
                           or interrupt the function]
      R←NUM2 'ENTER NUMERIC STRING:  '
ENTER NUMERIC STRING:   ←— [user presses "return"]
INVALID; PLEASE RE-TRY...
ENTER NUMERIC STRING:  Ø ← [user types O-U-T overstruck,
INVALID; PLEASE RE-TRY...   vainly trying to interrupt
ENTER NUMERIC STRING:       the function ...to no avail]
      |
  (...and on, ad infinitum...)
      ↓
```

...*The Moral:* while there may be some legitimate times where you want to "trap" a user's input without letting him interrupt the function, be sure that you use this sort of capability with discretion; don't make your functions unresponsive to the user.

Be aware that, using dyadic execute, you can now write uninterruptable functions. Be careful that you don't work yourself into a box.

Also take care to avoid name conflicts in functions that use execute (both monadic and dyadic). A user who is entering lots of repetitive data may wish to set up a variable in the workspace, and enter its name in response to the prompt for input. Fine, but with this particular function he would suddenly discover "mysterious" operations occuring if the name that he chose was "*T*" or "*Z*".

Ambi-valent defined functions

The "valence" of a function is a count of its explicit arguments: a monadic function has a valence of one, and a dyadic function has a valence of two. An ambi-valent function, then, is one which may be used with both valences.

A dyadic user-defined function may now be invoked either with or without its left argument. This allows you to write functions that more closely resemble the operation of primitive functions. Within the function, you must then check to see if the left argument was supplied before you reference its value; you can do this with *⎕NC* (name classification).

Here's a simple function for finding the n-th root of a number:

```
      ∇ Z←N ROOT A
    [1]   Z←A*÷N
      ∇

        2 ROOT 64 729 4096
    8 27 64

        (2 ROOT 64 729 4096)*2
    64 729 4096

        3 ROOT 64 729 4096
    4 9 16

        2 3 ROOT 16 125
    4 5
```

Perhaps a form that would be more convenient to use would be one which would use a common default value for the left argument if no value is supplied; let's assume, for instance, that we would usually use this for finding square roots. If the function is called <u>without</u> a left argument, the name used for the left argument ("*N*") would have no value. This can be checked using *⎕NC*, like this:

```
      ∇ Z←N ROOT A                                ∇ Z←N ROOT A
    [1]   →(0≠⎕NC 'N')/L2     ...or...      [1]   ⍎(0=⎕NC 'N')/'N←2'
    [2]   N←2                               [2]   Z←A*÷N
    [3]  L2:Z←A*÷N                                ∇
      ∇

        2 ROOT 64 729 4096
    8 27 64

        ROOT 64 729 4096
    8 27 64

        3 ROOT 64 729 4096
    4 9 16
```

Notice that *ROOT* now works with or without a left argument, and uses 2 as a default if the left argument is elided.

Ambi-valence can be used to supply a commonly-used value by default, as we did here, or it can supply an argument which would otherwise require cumbersome entry. For example, assume that you are using a *FIND* function which will look through selected functions whose names are listed in its left argument for a character string which is specified in its right argument. [An excellent example of such a function is offered in workspace 12 *EDITFN*.] To tell it that you want to look through all of the functions in the workspace it may be necessary to enter something like (⎕*NL* 3) *FIND* '*CHAR STRING*'. Most authors have long recognized common cases like this, and have provided a short-hand notation: '' *FIND* '*CHAR STRING*'. Now, using ambi-valence, the notation can be made one step easier: *FIND* '*CHAR STRING*'. Ambi-valence, then, is a useful tool for situations where you may frequently want to indicate "all values".

* * *

If you accidentally call a dyadic defined function without its left argument, it used to respond with a *SYNTAX ERROR*. Now it will invoke the function, and (if you haven't provided for it) will produce a *VALUE ERROR* the first time that the left argument is referenced. Recognizing this change may save you some time during trouble-shooting.

```
Previous response:              New response:

    ∇ Z←A PLUS B                    ∇ Z←A PLUS B
[1]   Z←A+B                     [1]   Z←A+B
    ∇                               ∇

      3 PLUS 5                        3 PLUS 5
8                               8

      PLUS 5                          PLUS 5
SYNTAX ERROR                    VALUE ERROR
      PLUS 5                    PLUS[1] Z←A+B
      ∧                                   ∧

      )SI                             )SI
                                PLUS[1] *

      PLUS                            PLUS
SYNTAX ERROR                    SYNTAX ERROR
      PLUS                            PLUS
      ∧                               ∧
```

Notice, in the last example, that ambi-valence does not allow the function to be called niladically.

⍋ and ⍒ have been upgraded

An improved sorting capability is now available through the use of dyadic grade. Previously, grade accepted only numeric vectors; sorting a character matrix could occur only by first encoding the character text into a numeric vector, and then grading that vector. Dyadic grade can now do this directly, by specifying the desired collating sequence as the left argument:

```
          M←4 4⍴'NOW IS  THE TIME'
          ALF←' ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'

          ALF⍋M
      2 1 3 4

          M[ALF⍋M;]
      IS
      NOW
      THE
      TIME
```

The left argument to grade can be whatever alphabet is best suited to your own application. For example, here is a sample text matrix sorted four different ways:

' ABC...XYZ'	' ABC...A̲B̲C̲...'	' AA̲BB̲CC̲DD̲...'	2 27⍴' ABC...XYZ' ' A̲B̲C̲...X̲Y̲Z̲'
AMA	AMA	AMA	AMA
PHOSPHATE	PHOSPHATE	A̲MA	A̲MA
PH̲	PH̲	A̲M̲A̲	A̲M̲A̲
P̲HILODENDRON	A̲MA	PHOSPHATE	PH̲
A̲MA	A̲M̲A̲	PH̲	P̲HILODENDRON
A̲M̲A̲	P̲HILODENDRON	P̲HILODENDRON	PHOSPHATE

Note that any of the sorting methods can take care of the obvious cases ...such as putting "AMA" first... but the cases rapidly become more complex when we introduce different fonts (both underscored and non-underscored characters) within the same matrix. In the fourth case, a matrix left argument was used, looking like this:

```
      ALF←2 27⍴' ABCDEFGHIJKLMNOPQRSTUVWXYZ A̲B̲C̲D̲E̲F̲G̲H̲I̲J̲K̲L̲M̲N̲O̲P̲Q̲R̲S̲T̲U̲V̲W̲X̲Y̲Z̲'
      ALF
 ABCDEFGHIJKLMNOPQRSTUVWXYZ
 A̲B̲C̲D̲E̲F̲G̲H̲I̲J̲K̲L̲M̲N̲O̲P̲Q̲R̲S̲T̲U̲V̲W̲X̲Y̲Z̲
```

This case says that our primary interest in sorting the text should be spelling... both fonts have the same weighting since the alphabetics in each row are aligned. But if several words have identical spelling ("AMA", "A̲MA", and "A̲M̲A̲"), then they should be sorted according to the row order of the fonts in the left argument. Thus, "AMA", "$\underline{A}MA$", and "$\underline{AMA}$" are grouped together in the final list.

If a character in the right argument doesn't appear at all in the left argument, it's treated in a fashion analogous to the action of *A*ι*B*... the unknown characters will be pushed to the end of the list, in "first-come-first-served" order:

```
                M←5 3ρ'○○○XXXAAA⎕⎕⎕∩∩∩'
                M['⎕○∩'⍋M;]
          ⎕⎕⎕
          ○○○
          ∩∩∩
          XXX
          AAA
```

For more information on the use of dyadic grade, refer to ACM's *APL*79 Conference Proceedings; "Sorting - a New/Old Problem", by Howard J. Smith, Jr. (Part 1, page 123).

Withdrawal of obsolete system facilities

Some system commands have been removed

The *)WIDTH*, *)ORIGIN*, and *)DIGITS* commands have been withdrawn. In their stead, you should use the following system variables:

System Command	System Variable
)WIDTH	⎕*PW* (printing width)
)ORIGIN	⎕*IO* (index origin)
)DIGITS	⎕*PP* (printing precision)

I-beams will be removed at year-end

Also be advised that the ancient I-beam functions will finally be withdrawn around the end of 1980. If you still have any of these old I-beams in your functions, please look into replacing them as soon as possible (don't get caught in a last-minute rush). If you need some assistance in doing this, please contact us.

I-beam	Approximate replacement	Purpose
⌶19	¯1↑⎕*AI*	Keyboard Unlock time
⌶20	3↓⎕*TS*	Time of day
⌶21	⎕*AI*[2]	CPU time used during this session
⌶22	⎕*WA*	Amount of workspace available
⌶23	⎕*UL*	User load
⌶24	⎕*AI*[3]	Session start time
⌶25	3↑⎕*TS*	Current Date
⌶26	1↑⎕*LC*	Current line number being executed
⌶27	⎕*LC*	Vector of line numbers in state indicator
⌶28	⎕*TT*	Terminal type
⌶29	1↑⎕*AI*	User number

↑
(origin 1 assumed throughout)

By "approximate replacement", we mean that the recommended expression yeilds roughly the same information, although it is typically in different units. The newer facilities are in generally much more "user-friendly" units than the I-beams were. For example, ⎕*TS* returns the current time and date as year-month-day-hour-minute-second-millisecond... ⌶20 gave the time in 60-ths of a second since the last midnight previous to your sign-on. It would therefore make sense to re-write the expression that they appear in rather than to convert the quads to old units, and then back to "friendly" units.

Heterogeneous output will be removed (date not yet set)

Also slated for removal in the near future is heterogeneous output, sometimes called mixed output. This was the old practice of printing both numeric and character data on the same line by separating them with semi-colons:

```
               N←127
               'HEIGHT IS ';N;' UNITS'
         HEIGHT IS 127 UNITS
```

A better approach is to format the numeric data into character data, like this:

```
               'HEIGHT IS ',(⍕N),' UNITS'
         HEIGHT IS 127 UNITS
```

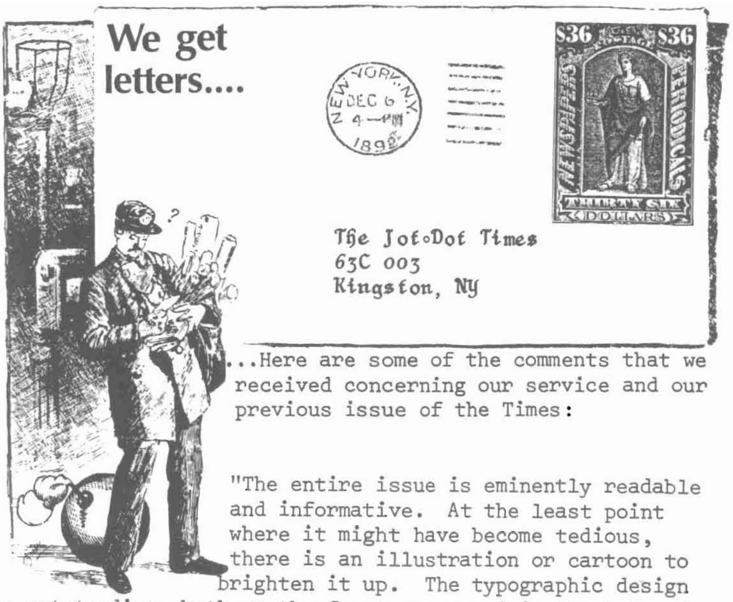
∆ and ⍙ will be removed as alphabetics (date not yet set)

Currently, the characters "∆" and "⍙" are allowed to be used as alphabetics; they currently have no other defined purpose. In the future, "_" and "¯" (respectively) will replace them as alphabetics. However, "_" and "¯" will not be allowed to start names (much like numerics). The system can do a conversion when necessary to replace embedded ∆ and ⍙, but leading characters will be a problem that you should probably consider approaching now.

* * *

If you do not already have a copy of the *APL* Language Manual (GC26-3847), you would be well advised to get yourself one. This manual defines what the *APL* language is supposed to do as opposed to what any particular implementation of the language does. Please realize that changes such as the ones that we have discussed here are not frivolous. Generally, they are made to allow for the future development of *APL*. To protect yourself from future changes, follow the language manual as closely as possible (with these updates for internal *APLSV* systems).

There are always a few coding constructions which do not appear in the *APL* Language Manual, but which will work (for the nonce). Don't get tricked into thinking that they are necessarily part of the language. Be advised that the things discussed in this article are not supported, and that there will come a morning when they won't work anymore.



...Here are some of the comments that we received concerning our service and our previous issue of the Times:

"The entire issue is eminently readable and informative. At the least point where it might have become tedious, there is an illustration or cartoon to brighten it up. The typographic design is outstanding, both on the front cover and in the body of the text. The humour is splendid-- all the more so because it is so unusual in an 'official' IBM publication. The size and format are just right. The reference information and articles are in the right order (phone numbers at the front, security guidelines at the back!).

"I have to admit failure. I cannot find ANY fault with it!

"Congratulations, you wordsmiths. The quality of the product reflects the efforts and skill that you put into it."

B. Martin
Endicott, NY

"Your publication is <u>W</u> (think German)."

A. Nunes
Kingston, NY

"Your Summer 1979 Issue was excellent... full of useful info, easy to read... how are you ever going to top it?"

A. Wolf
White Plains, NY

[*We're not.*]

First known example of one-liners

"Thank you for the article on "The One-Liner Syndrome". I've been criticized for years for not writing one-line programs. Now I have the satisfaction of knowing that I was right all along.

"You guys are great! How does the company let you get away with writing a newsletter that's not only useful but entertaining and witty besides?"

Name withheld by request
Kingston, NY

"I note with interest your castigation of the 'one-liner' fraternity. I agree, but I cannot understand why this obfuscacious bunch should have been entrusted with the responsibility and high honor of writing nearly all of our *APL* manuals.

"I'd like to see some new manuals, written with nearly the informality with which 'Jot∘Dot' was written. The emphasis should be on plain English with appropriate penalties for all authors who seem to be on the verge of exhibitionism."

A. Nires
Raleigh, NC

"More of the same! This publication is a model for efforts of this kind. It should be daily required reading for all authors of stuffy and stodgy communications...."

H. Clarke
Charlotte, NC

"Omit the newsletter-- put items in 1 *NEWS*."

D. Davis
Raleigh, NC

[**S*i*g*h*.... We tried that, somewhat unsuccessfully. After running two 50-line news items a month apart, we received mail telling us:*

"The length of these news items is absurd-- how about remembering that some of us still use 2741's!"

C. Martin
Kingston, NY

...*Sorry*.]

"I recently borrowed a copy of this excellent 'Times' from a friend who brought it from Rio. I think that you have heard many nice comments about this newsletter, therefore I will not take your time just saying how nice you are, but I can't go on without saying at least thank you!...."

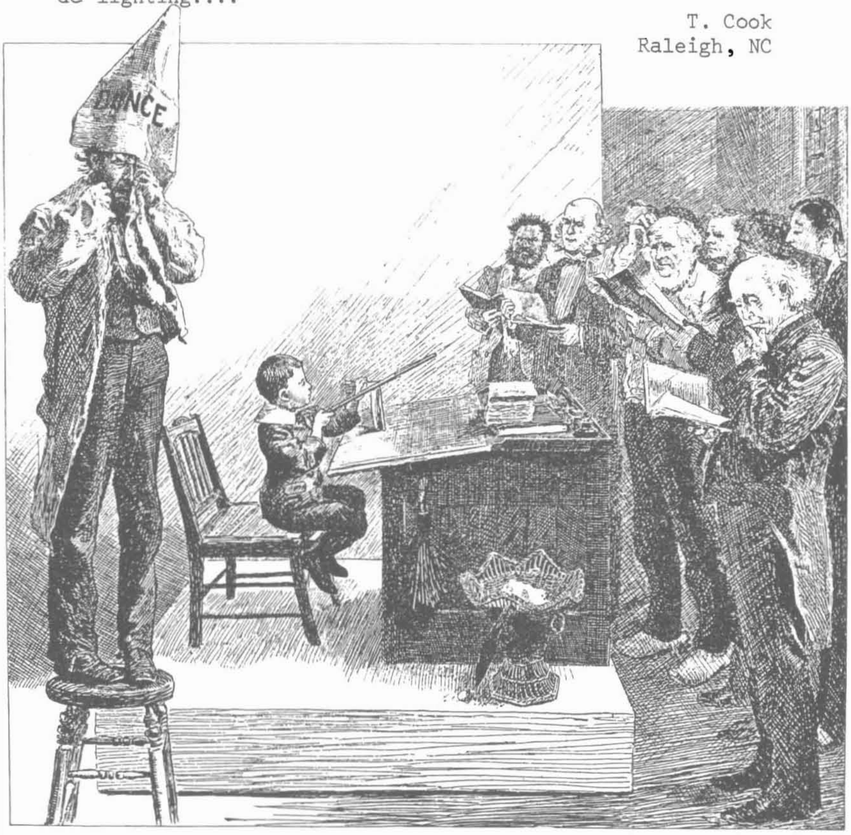
J. Elias
Sumare, Brazil

"Your 'Jot∘Dot Times' (and my son learning more *APL* in the local IBM Explorer group than I can remember) may just be the incentive that I needed to get back into *APL*. Thanks for a good issue."

D. Thompson
Kingston, NY

"How good is ∘.× ? ... ∘.× is so good that I keep a copy on my sailboat in the library created for windless days. Keeping this in mind, you must be careful that the material in the book does not change meaning when read under 12 volt dc lighting...."

T. Cook
Raleigh, NC



"Many thanks for the latest 'Jot∘Dot Times', and for providing a few chuckles to brighten the *APL* day. But please remove the dunce's cap (p. 27). Don Orth has got it wrong-- there <u>are</u> 441 inner products. There is a choice of 21 functions to the left of the dot, and for each of these is a further choice of 21 to the right, making 21×21 in all. This doesn't double-count ∧.∧ or any other."

N. Thomson
Hursley Park
Winchester, Hampshire, England

"My confidence in you guys has dropped considerably in the last month. Do you print <u>anything</u> without checking your facts? I wrote saying that there were 420 different inner products, not 441, <u>*and you printed it*</u>*!!* As so many of your readers have so kindly pointed out to me in the last month, there are indeed 441 different inner products.)"

D. Orth
Yorktown Heights, NY

"Your inner-product argument has been under discussion here in The *APL* Design Group. Proposed extensions to the *APL* language by Ken Iverson and myself have extended inner product so that it may be used with <u>any</u> dyadic functions. This includes mixed functions, derived functions, and user-defined functions. Now consider the question, "How many inner products are there?".

"Since there are an unlimited number of defined functions, the number of inner products is infinite! (If anyone is uncomfortable with such a large number, we could make every other one invalid.)

"Since the future contains so many inner products, I submit that we should lay to rest the controversy on the number of inner products in current *APL*.

"Don Orth claimed 420 different inner products; you claim 441. I propose that we settle on the average of these two numbers, 429.5, which can be easily computed as follows:

```
      T←420+441
      AVG←T÷2
    (I don't believe in one-liners)
```

"...I consider the case closed."

J. Brown
Yorktown Heights, NY

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

0 1 2 3 4 5 6 7 8 9

¨	dieresis	⍺	alpha	⍱	nor	∨	~
¯	overbar	⌈	upstile	⍲	nand	∧	~
<	less	⌊	downstile	⍒	del stile	∇	|
≤	not greater	_	underbar	⍋	delta stile	∆	|
=	equal	∇	del	⌽	circle stile	○	|
≥	not less	∆	delta	⍉	circle slope	○	\
>	greater	∘	jot	⊖	circle bar	○	-
≠	not equal	'	quote	⍟	log	○	⋆
∨	or	⎕	quad	⌶	I-beam	⊥	⊤
∧	and	(	left paren	⍫	del tilde	∇	~
-	bar	)	right paren	⍎	base null	⊥	∘
÷	divide	[	left bracket	⍕	top null	⊤	∘
+	plus	]	right bracket	⍀	slope bar	\	-
×	times	⊂	left shoe	⌿	slash bar	/	-
?	query	⊃	right shoe	⍝	cap null	∩	∘
⍵	omega	∩	cap	⍞	quote quad	⎕	'
∊	epsilon	∪	cup	!	quote dot	.	'
⍴	rho	⊥	base	⌹	domino	⎕	÷
~	tilde	⊤	top	$	dollar-sign	*S*	/
↑	up arrow	|	stile				
↓	down arrow	;	semicolon				
⍳	iota	:	colon				
○	circle	,	comma				
⋆	star	.	dot				
→	right arrow	\	slope				
←	left arrow	/	slash				
			space				

LM4: The *APL* Character Set

ACKNOWLEDGEMENTS

Most of the tables were taken from the standard *APL* manuals, and were reworked to reflect the current state of the internal version of *APLSV*. Each of these tables is marked with a key showing its source:

LM ↔ The *APL* Language Manual, GC26-3847-4
UG ↔ The *APLSV* Version 3 User's Guide, SH20-9087-0

For example, Figure 3 from The *APL* Language Manual is reprinted here as Figure LM3.

* * *

Thanks are in order to the entire Kingston *APL* Support group for their thoughts and suggestions in the preparation of this newsletter, and in particular, I extend many Thank-You's to Rhonda L. Johnson and Michael L. Higgs for their many hours of tedious table preparation and proofreading.

0	*IL*	32	⌊	64	∘	96	*K*	128	*P̲*	160	⍫	192	m	224	9
1		33	|	65	⎕	97	*L*	129	*Q̲*	161	*PFX*	193	n	225	0
2		34	∧	66	⍞	98	*M*	130	*R̲*	162	*HT*	194	o	226	(
3		35	∨	67	⍟	99	*N*	131	*S̲*	163	*UC IL*	195	p	227	)
4		36	<	68	⍲	100	*O*	132	*T̲*	164	*BS*	196	q	228	+
5		37	≤	69	⍱	101	*P*	133	*U̲*	165	§*RHLF*	197	r	229	-
6		38	=	70	⍝	102	*Q*	134	*V̲*	166	§*TL*	198	s	230	¹
7		39	≥	71	⍋	103	*R*	135	*W̲*	167	§*CE*	199	t	231	²
8		40	>	72	⍒	104	*S*	136	*X̲*	168	§*STP*	200	u	232	³
9		41	≠	73	⊖	105	*T*	137	*Y̲*	169	§*IRTN*	201	v	233	
10		42	⍺	74	⌿	106	*U*	138	*Z̲*	170		202	w	234	‰
11		43	∊	75	⍀	107	*V*	139	Δ̲	171		203	x	235	±
12		44	⍳	76	⌹	108	*W*	140	0	172		204	y	236	{
13		45	⍴	77	⍕	109	*X*	141	1	173		205	z	237	}
14	[	46	⍵	78	⍎	110	*Y*	142	2	174		206	&	238	¤
15	]	47	,	79		111	*Z*	143	3	175		207	¢	239	■
16	(	48	!	80		112	Δ	144	4	176		208	!	240	●
17	)	49	⌽	81		113	*A̲*	145	5	177		209	$	241	┌
18	;	50	⊥	82		114	*B̲*	146	6	178		210	#	242	└
19	/	51	⊤	83		115	*C̲*	147	7	179		211	%	243	┘
20	\	52	○	84		116	*D̲*	148	8	180	a	212	@	244	┐
21	←	53	?	85		117	*E̲*	149	9	181	b	213	|	245	┬
22	→	54	∼	86	*A*	118	*F̲*	150	.	182	c	214	¬	246	├
23		55	↑	87	*B*	119	*G̲*	151	_	183	d	215	"	247	┴
24		56	↓	88	*C*	120	*H̲*	152	*SPACE*	184	e	216	1	248	┤
25	¨	57	⊂	89	*D*	121	*I̲*	153	'	185	f	217	2	249	┼
26	+	58	⊃	90	*E*	122	*J̲*	154	:	186	g	218	3	250	┆
27	-	59	∩	91	*F*	123	*K̲*	155	∇	187	h	219	4	251	
28	×	60	∪	92	*G*	124	*L̲*	156	*CR*	188	i	220	5	252	
29	÷	61	_	93	*H*	125	*M̲*	157		189	j	221	6	253	
30	⋆	62	⍉	94	*I*	126	*N̲*	158	*UC BS*	190	k	222	7	254	
31	⌈	63	⌶	95	*J*	127	*O̲*	159	*LF*	191	l	223	8	255	

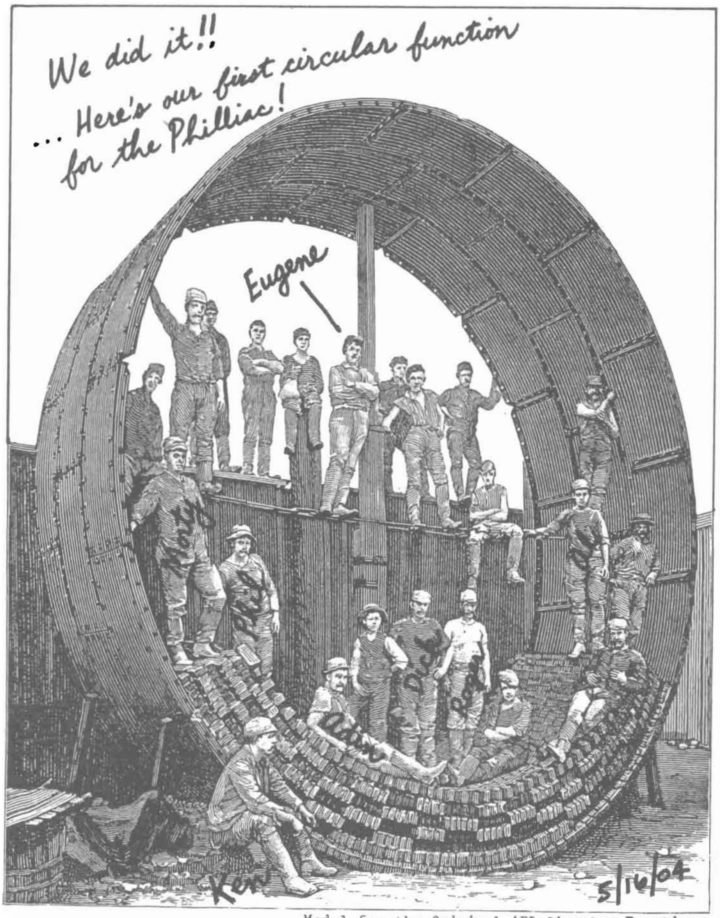
Legend: § Magnetic Card Selectric Terminal Controls

▨ 3270 characters only

The Magnetic Card Selectric Terminal Controls and the 3270 characters will only be generated for those devices.

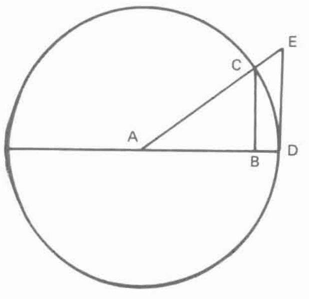
bs	backspace		il	idle		rhlf	reverse half-line feed
ce	card eject		irtn	index return		stp	stop
cr	carrier return		lf	line feed		tl	track link
ht	horizontal tab		pfx	prefix		uc	upper case

LM4: The Atomic Vector ⎕*IO*←0



Model for the Original *APL* Circular Functions
May 16th, 1904; Philadelphia

Courtesy of Jot Archives



```
AC=1
AB=0○BC
BC=0○AB
AE=4○DE
DE=¯4○AE
```

LM8: Pythagorean Functions

Monadic form f*B*		f	Dyadic form *A*f*B*	
Definition or Example	**Name**		**Name**	**Definition or Example**
+*B* is *B*	Conjugate	+	Plus	2+3.2 is 5.2
-*B* is 0-*B*	Negative	-	Minus	2-3.2 is ¯1.2
×*B* is (*B*>0)-*B*<0	Signum	×	Times	2×3.2 is 6.4
÷*B* is 1÷*B*	Reciprocal	÷	Divide	2÷3.2 is 0.625
|¯3.14 is 3.14	Magnitude	|	Residue	*A*|*B* is *B*-*A*×⌊*B*÷*A*+*A*=0
	Floor	⌊	Minimum	3⌊7 is 3
	Ceiling	⌈	Maximum	3⌈7 is 7
?*B* is Random choice from ⍳*B*	Roll	?	Deal	A Mixed Function (see Figure 10)
⋆*B* is (2.71828..)⋆*B*	Exponential	⋆	Power	2⋆3 is 8
⍟⋆*B* is *B* is ⋆⍟*B*	Natural logarithm	⍟	General logarithm	*A*⍟*B* is Log *B* base *A* *A*⍟*B* is (⍟*B*)÷⍟*A*
○*B* is *B*×3.14159...	Pi times	○	Circular, Hyperbolic, Pythagorean (see table at left)	
!0 is 1 !*B* is *B*×!*B*-1 or !*B* is Gamma(*B*+1)	Factorial	!	Binomial	*A*!*B* is (!*B*)÷(!*A*)×!*B*-*A* 2!5 is 10 3!5 is 10
~1 is 0 ~0 is 1	Not	~		
		∧	And	
		∨	Or	
		⍲	Nand	
		⍱	Nor	
		<	Less	Relations
		≤	Not greater	Result is 1 if the relation holds,
		=	Equal	0 if it does not:
		≥	Not less	3≤7 is 1
		>	Greater	7≤3 is 0
		≠	Not Equal	

B	⌊*B*	⌈*B*
3.14	3	4
¯3.14	¯4	¯3

(-*A*)○*B*	*A*	*A*○*B*
(1-*B*⋆2)⋆.5	0	(1-*B*⋆2)⋆.5
Arcsin *B*	1	Sine *B*
Arccos *B*	2	Cosine *B*
Arctan *B*	3	Tangent *B*
(¯1+*B*⋆2)⋆.5	4	(1+*B*⋆2)⋆.5
Arsinh *B*	5	Sinh *B*
Arcosh *B*	6	Cosh *B*
Artanh *B*	7	Tanh *B*

Table of Dyadic ○ Functions

A	*B*	*A*∧*B*	*A*∨*B*	*A*⍲*B*	*A*⍱*B*
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	1	0	0

LM6: Primitive Scalar Functions

Dyadic Function		Identity Element		Left-	Right
Plus	+	0		L	R
Minus	-	0			R
Times	×	1		L	R
Divide	÷	1			R
Residue	|	0		L	
Minimum	⌊	(note 1)		L	R
Maximum	⌈	(note 2)		L	R
Power	⋆	1			R
Logarithm	⍟			None	
Circle	○			None	
Binomial	!	1		L	
And	∧	1		L	R
Or	∨	0		L	R
Nand	⍲			None	
Nor	⍱			None	
Less	<	0	Apply	L	
Not greater	≤	1	for	L	
Equal	=	1	boolean	L	R
Not less	≥	1	arguments		R
Greater	>	0	only		R
Not equal	≠	0		L	R

Right, demonstrated

Notes:

1. The largest representable number.

2. The greatest in magnitude of representable negative numbers.

LM7: Identity Elements of Primitive Scalar Dyadic Functions

TYPE	Cause; CORRECTIVE ACTION
DEFN ERROR	Misuse of ∇ or ⎕ symbols: 1. The function is pendent. DISPLAY STATE INDICATOR AND CLEAR AS REQUIRED. 2. Use of other than a function name alone in reopening a definition. 3. Improper request for a line edit or display.
DOMAIN ERROR	Argument is outside the range of valid arguments (domain) of the function.
ENTRY ERROR	Invalid character has been transmitted or received.
⎕-- *IMPLICIT ERROR*	The system variable ⎕-- (for example, ⎕*IO*) has been set to an inappropriate value, or has been localized and not been assigned a value.
INDEX ERROR	Index value out of range.
INTERFACE QUOTA EXHAUSTED	Attempt to share more variables than allotted quota. REQUEST LARGER QUOTA FROM *APL* ADMINISTRATION (See page iv).
INTERRUPT	Execution was suspended within an *APL* statement. TO RESUME EXECUTION, ENTER A BRANCH TO THE STATEMENT INTERRUPTED.
LENGTH ERROR	Shapes not conformable.
NO SHARES	Shared variable facility not in operation.
NONCE ERROR	A syntactically correct statement has been entered, but cannot be executed because of an *APL* implementation restriction: the expression results in an error "for the nonce".
RANK ERROR	Ranks not conformable.
RESEND	Transmission failure. RE-ENTER. IF CHRONIC, REDIAL OR HAVE TERMINAL OR PHONE REPAIRED. IF YOU SUSPECT PHONE LINE PROBLEMS, CONTACT THE NETWORK LINE SERVICES GROUP (See page iv).
SI DAMAGE	The state indicator (an internal list of suspended and pendent functions) has been damaged in editing a function or in performing a)*COPY* or)*ERASE*..
SYMBOL TABLE FULL	Too many names used.)*SAVE*,)*CLEAR*,)*COPY* or)*SAVE*,)*CLEAR*,)*SYMBOLS*,)*COPY* or)*ERASE*,)*SAVE*,)*CLEAR*,)*COPY*
SYNTAX ERROR	Invalid syntax; e.g. two variables juxtaposed; function used without appropriate arguments as dictated by its header; unmatched parentheses.
SYSTEM ERROR	Fault in internal operation of the system. RELOAD. SEND TYPED RECORD, INCLUDING ALL WORK LEADING TO THE ERROR, TO *APL* PROGRAMMING SUPPORT (See page iv).
VALUE ERROR	Use of name that does not have a value. ASSIGN A VALUE TO THE VARIABLE OR DEFINE THE FUNCTION.
WS FULL	Workspace is filled (perhaps by temporary values produced in evaluating a compound expression, or by values of shared variables). CLEAR STATE INDICATOR, ERASE NEEDLESS OBJECTS, OR REVISE CALCULATIONS TO USE LESS SPACE.

LM3: Error Reports

Name	Sign[1]	Definition or Example[2]
<u>Functions Concerning the Structure of Arrays</u>		
Shape	ρA	ρP ↔ 4 ρE ↔ 3 4 ρ5 ↔ ι0
Reshape	VρA	Reshape A to dimension V 3 4ρι12 ↔ E 12ρE ↔ ι12 0ρE ↔ ι0
Ravel	,A	,A ↔ (×/ρA)ρA ,E ↔ ι12 ρ,5 ↔ 1
Reverse[3]	⌽A	DCBA ⌽X ↔ HGFE LKJI IJKL ⌽[1]X ↔ ⊖X ↔ EFGH ABCD ⌽P ↔ 7 5 3 2
Rotate[3]	V⌽A	3⌽P ↔ 7 2 3 5 ↔ ¯1⌽P BCDA 1 0 ¯1⌽X ↔ EFGH LIJK
Catenate, Laminate	A,A	P, ι2 ↔ 2 3 5 7 1 2 'T', 'HIS' ↔ 'THIS' P,[.5]P ↔ 2 3 5 7 2 3 5 7
Transpose,[4] dyadic	V⍉A	Coordinate I of A becomes coordinate V[I] of result AEI 2 1⍉X ↔ BFJ CGK DHL 1 1⍉E ↔ 1 6 11
Transpose, monadic	⍉A	Reverse order of coordinates ⍉E ↔ 2 1⍉E
<u>Functions Concerning Selection from Arrays</u>		
Take	V↑A	2 3↑X ↔ ABC ¯2↑P ↔ 5 7 EFG Take or drop |V[I]| first (V[I]≥0) or last (V[I]<0) elements of coordinate I
Drop	V↓A	2 3↓X ↔ L ¯2↓P ↔ 2 3

LM10 (Part 1 of 3): Primitive Mixed Functions

Name	Sign[1]	Definition or Example[2]
<u>Functions Concerning Selection from Arrays</u> (continued)		
Compress[3]	*V/A*	1 0 1 0/*P* ↔ 2 5
		1 3
		1 0 1 0/*E* ↔ 5 7
		9 11
		1 0 1/[1]*E* ↔ 1 2 3 4 ↔ 1 0 1⌿*E*
		9 10 11 12
Expand[3]	*V*\A	1 0 1\\⍳2 ↔ 1 0 2
		A BCD
		1 0 1 1 1*X* ↔ *E FGH*
		I JKL
Indexing[4] [5]	*V*[*A*]	*P*[2] ↔ 3
		P[4 3 2 1] ↔ 7 5 3 2
	M[*A*;*A*]	*E*[1 3;3 2 1] ↔ 3 2 1
		11 10 9
	A[*A*;...	*E*[1;] ↔ 1 2 3 4
	...;*A*]	*E*[;1] ↔ 1 5 9
		ABCD
		'*ABCDEFGHIJKL*'[*E*] ↔ *EFGH*
		IJKL
<u>Functions That Generate Selector Information</u>		
Index generator[4]	⍳*S*	First *S* integers
		⍳4 ↔ 1 2 3 4
		⍳0 ↔ an empty vector
Index of[4]	*V*⍳*A*	Least index of *A* in *V*, or 1+⍴*V*
		P⍳3 ↔ 2
		5 1 2 5
		P⍳*E* ↔ 3 5 4 5
		5 5 5 5
		4 4⍳4 ↔ 1
Membership	*A*∊*A*	⍴*W*∊*Y* ↔ ⍴*W*
		P∊⍳4 ↔ 1 1 0 0
		0 1 1 0
		E∊*P* ↔ 1 0 1 0
		0 0 0 0
Grade up,[4] monadic	⍋*V*	⍋3 5 3 2 ↔ 4 1 3 2
		The permutation which would order *V* (ascending or descending)
Grade down,[4] monadic	⍒*V*	⍒3 5 3 2 ↔ 2 1 3 4
Grade up,[4] dyadic	*A*⍋*A*	'*ABCDEFGHIJKLMNOPQRSTUVWXYZ*'⍋*Z* ↔ 5 2 3 1 4
		The left argument is an arbitrary alphabet showing the desired collating sequence; the right argument is the character array to be ordered
Grade down,[4] dyadic	*A*⍒*A*	'*ABCDEFGHIJKLMNOPQRSTUVWXYZ*'⍒*Z* ↔ 1 4 3 2 5
Deal[4]	*S*?*S*	*W*?*Y* ↔ Random deal of *W* elements from ⍳*Y*

LM10 (Part 2 of 3): Primitive Mixed Functions

Name	Sign[1]	Definition or Example[2]
<u>Functions That Involve Numerical Calculations</u>		
Matrix inverse	⌹*M*	⌹2 2⍴1 1 0 1 ↔ 1 ¯1 0 1 Arguments may be scalars, vectors, or matrices
Matrix division	*M*⌹*M*	(2 2⍴*P*)⌹2 2⍴1 1 0 1 ↔ ¯3 ¯4 5 7
Decode	*A*⊥*A*	10⊥1 7 7 6 ↔ 1776 24 60 60⊥1 2 3 ↔ 3723
Encode	*A*⊤*A*	24 60 60⊤3723 ↔ 1 2 3 60 60⊤3723 ↔ 2 3
<u>Functions That Involve Data Transformation</u>		
Execute, monadic	⍎*V*	⍎'1+2' ↔ 3 ⍎'*P*' ↔ 2 3 5 7
Execute, dyadic	*V*⍎*V*	'2+2'⍎'1+2' ↔ 3 '*P*'⍎'2+' ↔ 2 3 5 7
Format, monadic	⍕*A*	'¯1.5'∧.=⍕¯1.5 ↔ 1 ⍴⍕*E* ↔ 3 12 *X* ↔ ⍕*X*
Format, dyadic	*V*⍕*A*	4 1⍕*P* ↔ 2.0 3.0 5.0 7.0 4 ¯1⍕*P* ↔ 2*E*0 3*E*0 5*E*0 7*E*0
Format, picture	*V*⍕*A*	'$3,555.59 '⍕1234 100.749 ↔ $1,234.00 $100.75 (The left argument is a character vector, showing a picture of the intended output ...see page 40)

<u>Notes</u>:

1. Restrictions on argument ranks are indicated by: *S* for scalar, *V* for vector, *M* for matrix, and *A* for any array. See Figure LM11 for exceptions.

   Conformability requirements are given in The *APL* Language Manual where each function is defined.

2. Arrays used in examples:

```
        P                 E               X              Z
     2 3 5 7          1  2  3  4        ABCD           THE
                      5  6  7  8        EFGH           FUTURE
                      9 10 11 12        IJKL           IS
                                                       THE
                                                       ANSWER
```

3. The function is applied along the last axis; the symbols ⌿, ⍀, and ⊖ are equivalent to /, \, and ⌽, respectively, except that the function is applied along the first axis. In general, the relevant axis is determined by [*V*] or [*S*] after the function symbol.

4. Function depends on index origin.

5. Elision of any index selects all along that axis.

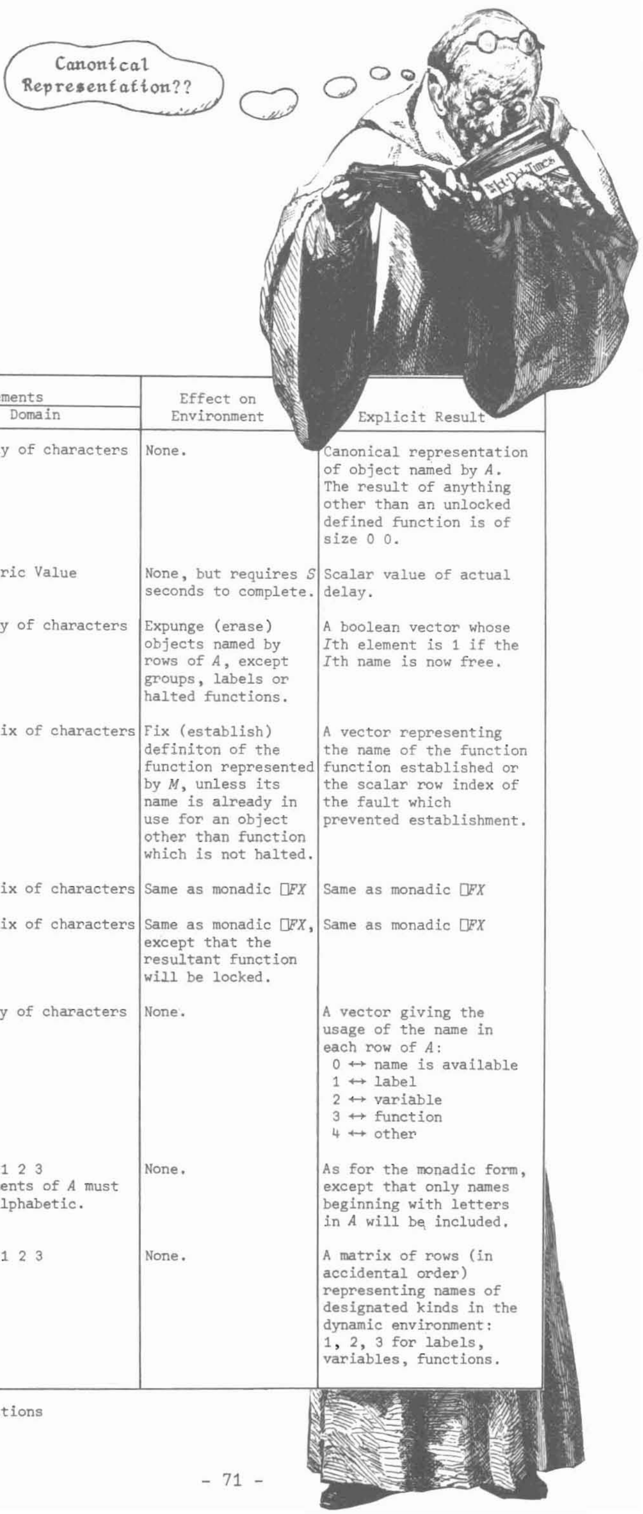
LM10 (Part 3 of 3): Primitive Mixed Functions

```
1. A scalar may be used in place of a one-element vector:

   a. as left argument of

          reshape            3⍴4          ↔   (,3)⍴4
          take               3↑⍳5         ↔   (,3)↑⍳5
          drop               3↓⍳5         ↔   (,3)↓⍳5
          expand             1\,5         ↔   (,1)\,5
          transpose          1⍉,5         ↔   (,1)⍉,5
          execute            'P'⍎'2+'     ↔   (,'P')⍎'2+'
          format             '5'⍕3        ↔   (,'5')⍕3
                           5⍕3.2 ↔ (,5)⍕3.2 ↔ 0 5 ⍕3.2

   b. as right argument of

          execute            ⍎'P'         ↔   ⍎,'P'
          branch             →4           ↔   →,4

2. A scalar is extended to conform as necessary:

   a. as left argument of

          compress           1 / ⍳3       ↔   1 1 1 /  ⍳3
          rotate             1⌽2 2⍴⍳4     ↔   1 1 ⌽ 2 2⍴⍳4

   b. as right argument of

          compress           1 0 1 / 2 ↔   1 0 1 / 2 2 2
          expand             1 0 1 \ 2 ↔   1 0 1 \2 2
          take               2 3 ↑3       ↔   2 3 ↑ 1 1⍴3

3. A one-element vector is permitted in place of a scalar:

   a. as left argument of

          compress           (,1)/⍳3         ↔   1/⍳3
          deal               (,3)?5          ↔   3?5
          rotate             (,2)⌽2 3 5 7 ↔   2⌽ 2 3 5 7

   b. as right argument of

          index generator    ⍳,5          ↔   ⍳5
          deal               3?,5         ↔   3?5
```

LM11: Scalar-Vector Substitutions for Mixed Functions

Type of Array	⍴*A*	⍴⍴*A*	⍴⍴⍴*A*
Scalar		0	1
Vector	*N*	1	1
Matrix	*MN*	2	1
3-Dimensional	*LMN*	3	1

LM12: Shape and Rank Vectors



Func-tion	Requirements		Effect on Environment	Explicit Result
	Rank	Domain		
⎕*CR A*	1≥ρρ*A*	Array of characters	None.	Canonical representation of object named by *A*. The result of anything other than an unlocked defined function is of size 0 0.
⎕*DL S*	0=ρρ*S*	Numeric Value	None, but requires *S* seconds to complete.	Scalar value of actual delay.
⎕*EX A*	2≥ρρ*A*	Array of characters	Expunge (erase) objects named by rows of *A*, except groups, labels or halted functions.	A boolean vector whose *I*th element is 1 if the *I*th name is now free.
⎕*FX M*	2=ρρ*M*	Matrix of characters	Fix (establish) definiton of the function represented by *M*, unless its name is already in use for an object other than function which is not halted.	A vector representing the name of the function function established or the scalar row index of the fault which prevented establishment.
0 ⎕*FX M*	2=ρρ*M*	Matrix of characters	Same as monadic ⎕*FX*	Same as monadic ⎕*FX*
1 ⎕*FX M*	2=ρρ*M*	Matrix of characters	Same as monadic ⎕*FX*, except that the resultant function will be locked.	Same as monadic ⎕*FX*
⎕*NC A*	2=ρρ*A*	Array of characters	None.	A vector giving the usage of the name in each row of *A*: 0 ↔ name is available 1 ↔ label 2 ↔ variable 3 ↔ function 4 ↔ other
A ⎕*NL N*	1≥ρρ*N*	∧/*N*∊1 2 3 Elements of *A* must be alphabetic.	None.	As for the monadic form, except that only names beginning with letters in *A* will be included.
⎕*NL N*	1≥ρρ*N*	∧/*N*∊1 2 3	None.	A matrix of rows (in accidental order) representing names of designated kinds in the dynamic environment: 1, 2, 3 for labels, variables, functions.

LM13: System Functions

Name	Value In Clear Ws	Meaningful Range	Purpose
⎕*AI*			Account information: identification, computer time, connect time, keying time (all times in milliseconds and cumulative during session)
⎕*AV*			Atomic vector: all 256 characters supported under *APLSV*, sorted in order of their internal representation codes
⎕*CT*	1*E*¯13	0 → 1	Comparison tolerance: used in monadic ⌈ ⌊, dyadic < ≤ = ≥ > ≠ ∊ ⍳
⎕*FC*	.,*0_	Characters	Format control: used in all forms of dyadic ⍕
⎕*HT*	⍳0	Positive integers and zero	Horizontal tab settings: a vector of integers or a scalar
⎕*IO*	1	0 or 1	Index origin: used in indexing and in *?* ⍳ ⍋ ⍒ ⍉ ⎕*FX*
⎕*LC*	⍳0		Line counter: statement numbers of functions in execution or halted, most recently activated first
⎕*LX*	''	Characters	Latent expression: executed upon activation of workspace
⎕*PP*	10	⍳16	Printing precision: affects numeric output and monadic ⍕
⎕*PW*	120	30 → 390	Printing width: affects all but bare output (⍞←), error reports, and messages
⎕*RL*	7*5	⍳¯2+2*31	Random link: used in *?*
⎕*TS*			Time stamp: year, month, day (of month), hour (on 24-hour clock), minute, second, millisecond
⎕*TT*			Terminal type: 0 ↔ Indeterminate or Phantom processor (such as *DEFER*) 1 ↔ Correspondence encoding 2 ↔ PTTC/BCD encoding 3 ↔ 1050, no longer supported 4 ↔ 3270-DAF 5 ↔ 3270 without *APL* feature, not currently being used
⎕*UL*			User Load
⎕*WA*			Working area available (in bytes)

LM14: System Variables

Comparison tolerance, ⎕*CT*	1*E*¯13
Format control, ⎕*FC*	.,*0_
Horizontal tabs, ⎕*HT*	Empty
Index origin, ⎕*IO*	1
Line counter, ⎕*LC*	Empty
Latent expression, ⎕*LX*	Empty
Printing precision, ⎕*PP*	10
Printing width, ⎕*PW*	120
Random link, ⎕*RL*	16807
Work area available, ⎕*WA*	Depends upon the local installation
State indicator	Cleared
Symbol table size	256
Workspace name	None (*CLEAR WS*)
Workspace password	None

LM22: Environment within a Clear Workspace

Form	Purpose	Normal Response	Trouble Reports
Active Workspace - Action Commands			
)*CLEAR*	Activate a clear ws	*CLEAR WS*	5
)*SYMBOLS* number	Set size of symbol table	*WAS* number	5
)*ERASE* [names]	Erase global objects named from active ws		5,12,25
)*COPY* wsid [pass]	Copy all global objects from named ws into active ws	*SAVED* time date	1,3,4,5,17, 25,26,28,29, 30
)*COPY* wsid [pass] names	Copy global objects named from designated ws into active ws	*SAVED* time date	1,3,4,5,12, 17,25,26,28, 29,30
)*PCOPY* wsid [pass]	Copy all objects from designated ws not named in active ws	*SAVED* time date	1,3,4,5,11, 17,25,26,28, 29,30
)*PCOPY* wsid [pass] names	Copy objects designated that are not named in active ws	*SAVED* time date	1,3,4,5,11 12,17,25,26, 28,29,30
)*GROUP* names	Gather objects into (or disperse) a group (first name designates group)		5,13,26,28
Active Workspace - Inquiry Commands			
)*SYMBOLS*	Give maximum number of names in ws	*IS* n, n *IN USE*	5
)*FNS* [alphabetic]	List defined functions functions (whose initials follow given character in alphabet)	[names]	5
)*VARS* [alphabetic]	List global variables (whose initials follow given character in alphabet)	[names]	5
)*GRPS* [alphabetic]	List groups (whose initials follow given character of alphabet)	[names]	5
)*GRP* name	List members of named group	[names]	5,26
)*SI*	List halted functions	state-indicator	5
)*SINL*	List halted functions and associated local names	state-indicator with name-list	5

LM19 (Part 1 of 4): System Commands

Form	Purpose	Normal Response	Trouble Reports
<u>Workspace Storage and Retrieval - Action Commands</u>			
)*WSID* wsid [newpass]	Change identification of active ws	*WAS* wsid	5
)*SAVE* wsid [newpass]	Replace named ws by copy of active ws	time date	2,4,5,14, 15,16,28
)*SAVE*	Place copy of active ws in library	time date wsid	4,5,14,16,28
)*CONTINUE*	Replace ws *CONTINUE* by copy of active ws and end use of *APL*	[time date *CONTINUE*] header; account	2,5,27,28
)*CONTINUE HOLD*	Replace ws *CONTINUE* by copy of active ws and end use of *APL*, but hold telephone connection	[time date *CONTINUE*] header; account	2,5,27,28
)*LOAD* wsid [pass]	Activate copy of named ws	*SAVED* time date	1,4,5,28,29
)*DROP* accessible-wsid	Drop ws from library	time date	4,5,30
<u>Workspace Storage and Retrieval - Inquiry Commands</u>			
)*WSID*	Give wsid of active ws	[number] name	5
)*LIB* [number] [letter]	List workspaces in designated library (whose initials follow given character in alphabet)	[names]	4,5
<u>Access to the System</u>			
)number:password	Identify user and start use of *APL*	["hi"-message] header; system [*SAVED* time date]	6,9,18,19, 20,22
)*OFF*	End use of *APL*	header; account	5,27
)*OFF HOLD*	End use of *APL*, but hold telephone connection	header; account	5,27
)*CONTINUE*	Replace ws *CONTINUE* by copy of active ws and end use of *APL*	[time date *CONTINUE*] header; account	2,5,27,28
)*CONTINUE HOLD*	Replace ws *CONTINUE* by copy of active ws and end use of *APL*, but hold telephone connection	[time date *CONTINUE*] header; account	2,5,27,28

LM19 (Part 2 of 4): System Commands

Form	Purpose	Normal Response	Trouble Reports
Administrative and Miscellaneous			
)	"Blot" (obfuscate) next input line	(blot pattern)	
★)*ATTACH* printer-number	Attach a 32xx-printer during a CON/370 session	*PRINTER ATTACHED*	5,10,23
★)*DETACH*	Detach your 32xx-printer during a CON/370 session	*PRINTER DETACHED*	5,27
)*PORT* initials	List port number[s] of user[s] currently signed on with indicated initials	port, initials	5
)*PORT* port	List all port numbers equal to or greater than the given port	port, initials	5
)*PORTS*	List all port numbers	port, initials	5
)*MSG OFF*	Prevent reception of any messages		5
)*MSG ON*	Restore reception of messages, and reprint last public-address message (if any)	[text]	5
)*MSG* port [text...]	Send a message to user at "port", and lock keyboard until reply is received	*SENT*	5,7
)*MSGN* port [text...]	Send a message to user at "port", with no reply expected	*SENT*	5,7
)*OPR* [text...]	Send a message to *APL* system operator, and lock keyboard until reply is received	*SENT*	5,7
)*OPRN* [text...]	Send a message to *APL* system operator, with no reply expeeted	*SENT*	5,7
)*PASSWORD*	Display date on which the current sign-on password will expire (preventing sign-on)	*EXPIRES* date	4,5
)*PASSWORD* old:new	Change the sign-on password to designated new password; old and new must be different, and must meet length rules (see page 14)	*EXPIRES* date	4,5,8,21

LM19 (Part 3 of 4): System Commands

★ ↔ *KINGSTON LOCAL MODIFICATION*

Form	Purpose	Normal Response	Trouble Reports
Administrative and Miscellaneous (continued)			
)QUOTA	Display administrative information and quotas (sample): ...workspace quota shared-variable quota CPU time limit (0=no limit) number of wss currently saved phone-line access-code, user name	10 8 .0 5 *XJDOE*	4,5

Notes:

1. Items in brackets are optional.

2. Abbreviations and meanings:

account	gives the correct time and compute time since last start and since beginning of the accounting period.
header	a port number, time of day, date, and user-code
"hi"-message	a message to users of the system at sign-on time
initials	the first three characters of the name assigned to an account
newpass	a password, which does not have to match a previous password
pass	a colon possibly followed by a password
port	a port number of a currently signed-on user
text	any one line of text, up to 120 characters long
ws	workspace
wsid	a ws name possibly preceded by a library number

3. The commands *)ERASE*, *)FNS*, and *)VARS* have variants that are system functions (see ⎕*EX* and ⎕*NC*).

LM19 (Part 4 of 4): System Commands

No	*TROUBLE REPORT*	Meaning	Remedy
1	*CLEAR WS*	The ws named in the)*LOAD*,)*COPY* or)*PCOPY* command is damaged due to system problems	Contact *APL* System Support (see page iv)
2	*COMMAND DISALLOWED*	Attempted rename of a restricted (load-only) workspace	
3	*DEFN ERROR*	Attempted copy or protected copy of function definition as response to ⎕ input request	
4	*IMPROPER LIBRARY REFERENCE*	1. Number is not a library number, or 2. Attempted save into alien library, or 3. Attempted reference to alien *CONTINUE* ws, or 4. Attempted copy from a restricted (load-only) ws	
5	*INCORRECT COMMAND*		
6	*INCORRECT SIGN-ON*		
7	*MESSAGE LOST*	Terminal was interrupted before message was received by other end	
8	*NEW PASSWORD UNACCEPTABLE*	Password does not meet requirements	(see page 14)
9	*NO PORTS AVAILABLE*	All lines into *APL* are busy	
10	*NO PRINTER ATTACHED*	32xx printer was attached earlier in this session, and not detached	Issue)*DETACH* first
11	*NOT COPIED*: names	Global homonyms in active ws are protected	
12	*NOT FOUND*: purported names	WS does not contain global objects with purported names	

LM20 (Part 1 of 3): Trouble Reports

No	*TROUBLE REPORT*	Meaning	Remedy
13	*NOT GROUPED, NAME IN USE*	First name is name of a global function or variable	1. Use different name for group, or 2. Erase global object if not needed
14	*NOT SAVED, THIS WS IS CLEAR WS*	A clear ws has no name and cannot be stored	
15	*NOT SAVED, THIS WS IS* wsid	Attempted replacement of a stored ws whose identification does not match that of the active ws	Remove stored ws, then store active ws
16	*NOT SAVED, WS QUOTA USED UP*	Alloted number of stored ws's previously reached	1. Drop an unneeded ws, or 2. Ask *APL* Administration to increase quota (see page iv)
17	*NOT WITH OPEN DEFINITION*	This particular command may not be issued from within function editor	Close function definition, then re-try
18	*NUMBER IN USE*		Consult *APL* operator (see page iv)
19	*NUMBER LOCKED OUT*	Authorization for use of number has been withdrawn (possibly due to non-usage)	Consult *APL* Administration (see page iv)
20	*NUMBER NOT IN SYSTEM*	1. Number entered is not an account number, or 2. Password missing, or 3. Wrong password used	Consult *APL* Administration (see page iv)
21	*OLD PASSWORD INCORRECT*	Password entered does not match your current sign-on password	
22	*PASSWORD EXPIRED*	Too long since sign-on password was changed (see page 14)	Consult *APL* Administration (see page iv)

LM20 (Part 2 of 3): Trouble Reports

No	*TROUBLE REPORT*	Meaning	Remedy
23	*PRINTER NOT FOUND*	Invalid 32xx printer number given	
24	*RESEND*	Transmission failure; one or more characters were garbled during transmission	If chronic, redial or have terminal or phone repaired. If you suspect phone-line problems, call Network Line Services (see page iv).
25	*SI DAMAGE*	State indicator damaged while performing an)*ERASE* or)*COPY* command	
26	*SYMBOL TABLE FULL*	Too many names used	Erase objects not needed, save ws, clear active ws, and perhaps change limit, using)*SYMOBLS*, copy the saved ws and rename active ws
27	*TERMINAL MUST BE IN DISPLAY MODE*		
28	*WS FULL*	Workspace full, possibly because of 1. Temporary values produced during evaluation of an expression, or 2. Value assigned to shared variable by partner	1. Erase objects not needed, or 2. Clear state indicator, or 3. Revise method of calculation
29	*WS LOCKED*	1. Password missing, or 2. Wrong password used, or 3. Ws is not locked, but a password was used in the command.	
30	*WS NOT FOUND*	No stored ws with given identification	

LM20 (Part 3 of 3): Trouble Reports

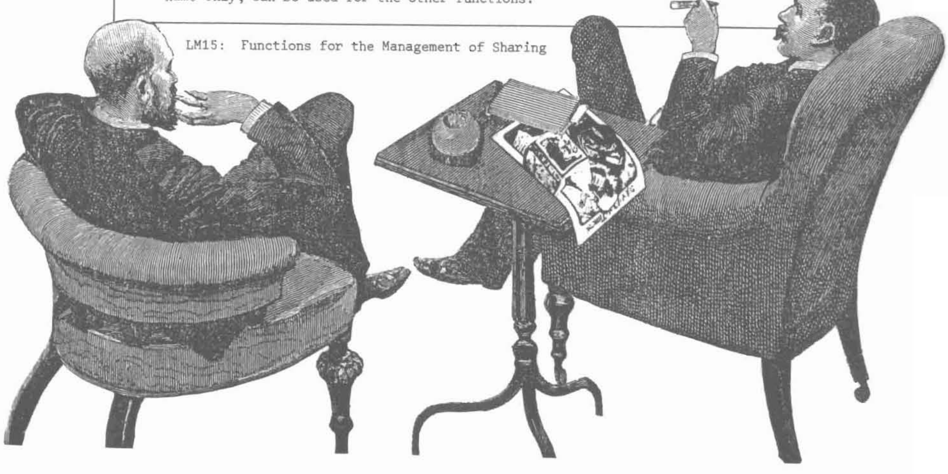
Function	Requirements (1)			Effect on Environment	Explicit Result
	Rank	Length	Domain		
P ⎕*SVO* *N* (for TSIO, *P* is 370)	2≥ρρ*N*	(×/ρ*P*)∊1,¯1↓ρ*N*	*P*∊¯1+ι2*31 (2)	Tenders offer to processor *P* if first (or only) name of a pair is not previously offered and not already in use as the name of an object other than a variable. The offer is general (to anyone) if 0=*P*	Degree of coupling now in effect for the name pair. Dimension: ,×/¯1↓ρ*N*
⎕*SVO* *N*	2≥ρρ*N*	None	(2)	None	Degree of coupling now in effect for the name pair. Dimension: ,×/¯1↓ρ*N*
C ⎕*SVC* *N*	2≥ρρ*N* 2≥ρρ*C*	(1≥ρρ*C*)∧1=×/ρ*C* or (ρ*C*)=(¯1↓ρ*N*),4	∧/*C*∊0 1 (2)	Sets access control	New setting of access control. Dimension: (¯1↓ρ*N*),4.
⎕*SVC* *N*	2≥ρρ*N*	None	(2)	None	Existing access control
⎕*SVR* *N*	2≥ρρ*N*	None	(2)	Retracts offer (ends sharing)	Degree of coupling before this retraction. Dimension: ,×/¯1↓ρ*N*
⎕*SVQ* *P*	1≥ρρ*P*	1≥ρ,*P*	*P*∊¯1+ι2*31	None	If 0=ρ*P*: Vector of identifications of processors making offers to this user If 1=×/ρ*P*: Matrix of names offered by processor *P* but not yet shared

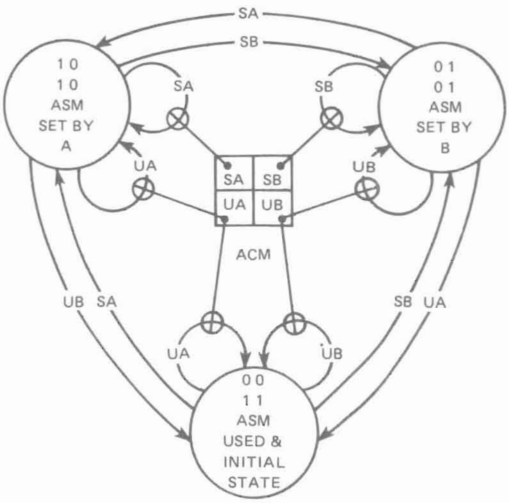
Notes:

1. If a requirement is not met the function is not executed and a corresponding error report is printed.

2. Each row of *N* (or *N* itself if 2≥ρρ*N*) must represent a name or pair of names. If a pair of names is used for an offer (dyadic ⎕*SVO*), either the pair, or the first name only, can be used for the other functions.

LM15: Functions for the Management of Sharing





Legend:

SA SB UA UB: Denote *set* or *use* by A or B.
ACM: Access Control Matrix
ASM: Access State Matrix

A one in an element of ACM inhibits the associated access. Allowable accesses are given by the zeros in $ACM \wedge ASM$. Access control vectors as seen by A and B, respectively, are $,ACM$ and $,\phi ACM$.

The access state matrix represents the last access: ones occur in the last row if it is not a set, and in a column if it is, the first column if set by A and the last if set by B.

LM16: Access Control of a Shared Variable

Access Control Vector as seen by		Comments
A	B	
0 0 0 0	0 0 0 0	No constraints.
0 0 1 1	0 0 1 1	Half-duplex. Ensures that each use is preceded by a set by partner.
1 1 0 0	1 1 0 0	Half-duplex. Ensures that each set is preceded by an access by partner.
1 1 1 1	1 1 1 1	Reversing half-duplex. Maximum constraint.
0 1 1 0	1 0 0 1	Simplex. Controlled communication from B to A (for card reader, etc.).

UG17: Some Useful Settings for the Access Control Vector

ABBREV.	OPERATION	MEANING AND USAGE
SW	Sequential Write	Create a new data set, and rewrite or append to an existing data set.
SR	Sequential Read	Read records sequentially from an existing data set.
IRW	Indexed Read and Write	Read and write records in arbitrary sequence from or to an existing data set.
IR	Indexed Read	Read records in arbitrary sequence from an existing data set.
RENAME	Rename	Change the name of an existing data set.
DELETE	Delete	Delete an existing data set.
IC	Indirect Command	Execute a prepared command from a command data set.

UG5: Data Management Operations

DISCRIMINANT		
VALUE	NAME	USER QUALIFICATION
8	SPACE	Allowed to create new direct access data sets, which implies the allocation of storage space.
4	DEVICE	Allowed to use the *UNIT* and *VOLUME* parameters, which implies the allocation of specific devices or storage units, as well as the *CATALOG* and *UNCATALOG* parameters, described in the section "System Level Operations" of the *APLSV* Version 3 User's Guide.
2	ACCESS	Allowed indexed access or sequential reading of other TSIO users' non-reserved data sets, given a knowledge of their identification.
1	SYSTEM	Has use of commands beyond the seven available to all users, and access to any data set, including those vital to system operation, within the constraints of the operating system security provisions.
On the the Kingston system, Level 10 (SPACE and ACCESS) is given to all users.		

UG6: User Levels

PARAMETER USAGE	COMMENTS
BLKSIZE=block size	Required with *DISP=NEW* or *LABEL=NL* or *BLP*
CODE= $\left\{\begin{matrix}A\\B\\C\\E\\F\\I\\S\end{matrix}\right.$	Required when accessing non-*APL* data sets
★ *COPIES*=n	Number of copies of output from *SYSOUT*; "n" may be from 0 to 255
DEN= $\left\{\begin{matrix}0\\1\\2\\3\\4\end{matrix}\right.$	Used only with *UNIT*=tape and *DISP=NEW*
DISP=($\left\{\begin{matrix}NEW\\OLD\\MOD\\SHR\end{matrix}\right\}$ [, $\left\{\begin{matrix}LEAVE\\REREAD\end{matrix}\right\}$])	See Figure UG7 for defaults
$\left.\begin{matrix}DSN\\DSNAME\end{matrix}\right\}$=⌈[-]user number]dsname[(member)]	Required except for *READVTOC*
EXPDT=yyddd	Only with *SW DISP=NEW*
KEYLEN=keylength	Only with *DISP=OLD* or *SHR* and *UNIT*=direct access
LABEL=(data set seq no.[, $\left\{\begin{matrix}BLP\\NL\\SL\end{matrix}\right\}$])	Only with *UNIT*=tape
LRECL=record size	With *RECFM=FB* and *DISP=NEW* or *LABEL=BLP* or *NL*
NEWNAME=⌈[-]user number]dsname[(member)]	With *RENAME* command

UG8 (Part 1 of 2): TSIO Command Parameters

PARAMETER USAGE	COMMENTS
RECFM= { *U* / *F* / *V* / *FB* / *VB* / *FBS* } { [*A*] / *M* }	Only with *DISP=NEW* or *LABEL=BLP* or *NL*
★ *SC*= { *UN* / *IU* / *IC* / *IR* }	Security Class for *SYSOUT*: Unclassified IBM Internal Use IBM Confidential IBM Confidential-Restricted
★ *SI*= { *H* / *S* }	Special instructions for *SYSOUT*: Hold for pick-up Special handling (")*LOAD* 1 *AIDS*" and type "*SYSOUT*" to send instructions)
SPACE=(blocklength,(primary [,secondary[,directory]])[,*RLSE*])	"*RLSE*" releases unused space when dataset is closed following initial creation only
SYSOUT=(output class[,[program name] [,form number]])	Consult local Programming Support (see page iv)
TRTCH= { *C* / *E* / *T* / *ET* }	Seven track tape only Normally not required
UNIT= { device type / group name / unit address }	
★ *USER*=account number	Send *SYSOUT* to another user of this system
{ *VOL* / *VOLUME* } =[*SER*=]volume serial	Use prevents cataloging a non-TSIO data set

UG8 (Part 2 of 2): TSIO Command Parameters

★ ↔ ***KINGSTON LOCAL MODIFICATION***

	DSN	*DISP* (2)	*NEWNAME*	*CODE*	*RECFM* (1)	*LRECL* (1)	*BLKSIZE* (1)	*SPACE* (2)	*SYSOUT* (2)
SW New	R	R		O *A*	O *U*	R for blocked data	R	O (4)	E
Not New	R	O *OLD*		O *A*	C L	C L	C L		E
Tran-sitory	E			O *E*	O *VB*	O 137	O 689	O (4)	R
SR	R	O *SHR*		Q *A*	O L	O L	O (3) L		E
IR	R	O *SHR*		Q *A*	C	C	C		E
IRW	R	O *OLD*		Q *A*	C	C	C		E
RENAME	R	C *OLD*	R						
DELETE	R	C *OLD*			O		O		
IC	R	E	E	E	E	E	E	E	E

LEGEND

USE

C	With caution
E	Error
O	Optional
R	Required
Q	Required for *CODE≠A*
Blank-Ignored	

DEFAULT

L From label or record

Parameter values (in *APL* font)

OPERATIONS

SW	Write sequentially
SR	Read sequentially
IR	Read with index
IRW	Read and write with index
RENAME	Change the data set name
DELETE	Expunge the data set
IC	Command indirectly

NOTES

1. See Figure UG15.
2. Parameter value may be compound.
3. *BLKSIZE* is required for data sets that do not have standard labels.
4. (1000,(100,0)) for sequential data set, (1000,(100,0,5)) for partitioned data set.

UG7: Data Set Operations, Parameter usage, and Default Values

APL EBCDIC	*A B C D E F G H I J K L M N O P Q R S T U V W X Y Z* A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
APL EBCDIC	*A̲ B̲ C̲ D̲ E̲ F̲ G̲ H̲ I̲ J̲ K̲ L̲ M̲ N̲ O̲ P̲ Q̲ R̲ S̲ T̲ U̲ V̲ W̲ X̲ Y̲ Z̲* a b c d e f g h i j k l m n o p q r s t u v w x y z
APL EBCDIC	0 1 2 3 4 5 6 7 8 9 space 0 1 2 3 4 5 6 7 8 9 space
APL EBCDIC	- ~ * ¨ ' , . ; : ( ) / \ ! ÷ < > *$̸* α Δ = ∧ ? + _ | ϕ ⊖ - ¬ * " ' , . ; : ( ) / \ ! % < > $ @ # = & ? + _ | +0 -0

<u>Notes</u>:

1. All other *APL* characters map into EBCDIC space, except *APL* ← ¯ ≠ which map, respectively, into EBCDIC = - $.
2. All other EBCDIC characters map into *APL* ∘.
3. The characters $ @ # are referred to as <u>national</u> characters and may be represented by different graphics in different countries.
4. The +0 and -0 are obtained by overpunching. These values occur in the zoned decimal numeric representation used by keypunch data.

UG9: *APL*-EBCDIC and EBCDIC-*APL* Translation (TSIO *CODE=E*)

RECFM	Description, transmission form
F	Fixed length blocks, containing only one record, transmitted as a vector. Only format available for creating datasets for use with *IR* and *IRW*, or *CODE=A*.
FB	Variable length blocks, of length up to *BLKSIZE*, each containing some number of fixed length records; each record transmitted as a vector.
FBS	Fixed length blocks, each containing a fixed number of records; each block transmitted as a <u>matrix</u> of size *M N* where *M* is *BLKSIZE* ÷ *LRECL* and *N* is given below. *N* is *LRECL* ÷ .125 1 1 4 8 when *CODE* is *B* *C* *E* *I* *F* Final record may be of size *L N* where *L* < *M*; next return code is from dataset close. For efficiency and ease of use, *FBS* is preferred over *FB*.
VB	Variable length self-describing blocks, containing variable length self-describing records, which are transmitted as a sequence of vectors whose length, when writing, must be less than *LRECL*-4 to accommodate the descriptors. *BLKSIZE* must be at least *LRECL* + 4.
V	Exactly the same as *VB* except that, when writing, a block is sent to the storage medium each time a record is transmitted, rather than only when the next record to be transmitted will not fit in the current block.
U	Variable length blocks each consisting of one record, transmitted as a vector. Can be used to read data in any other format, so long as *BLKSIZE* is large enough.

UG10: Parameters Related to Format

ACTION (1)	CONTROL CHARACTERS (2)		
	A-ANSI		M-MACHINE
	For *CODE=C* use ⎕*AV* indexed by	For *CODE=E* or *CODE=S* use *APL* character	For *CODE=C* use ⎕*AV* indexed by
Suppress line spacing	78 (5)	+	1 (5)
Space one line	64	space	9
Space two lines	240	0	17
Space three lines	96	-	25
Skip to Channel 1	241	1	137
Skip to channel 2	242	2	145
Skip to channel 3	243	3	153
Skip to channel 4	244	4	161
Skip to channel 5	245	5	169
Skip to channel 6	246	6	177
Skip to channel 7	247	7	185
Skip to channel 8	248	8	193
Skip to channel 9	249	9	201
Skip to channel 10	193	*A*	209
Skip to channel 11	194	*B*	217
Skip to channel 12	195	*C*	225
Select punch pocket 1 (3)	229	*V*	1
Select punch pocket 1, CB (4)			33
Select punch pocket 2	230	*W*	65
Select punch pocket 2, CB			97

Notes:
1. Action takes place before line is printed for ANSI, and after line is printed for machine encoding.
2. Characters other than these default to space one line or select punch pocket 1, as appropriate.
3. Characters shown for 2520 Punch. See OS/VS references for others.
4. CB: column binary.
5. Zero-origin indices.

UG11: Unit Record Output Control Characters

	2314		3330/3330-II		3340/3344		3350	
Blocks per Track	Bytes per Block	% Utili-zation	Bytes per Block	% Utili-zation	Bytes per Block	% Utili-zation	Bytes per Block	% Utili-zation
1	7294	100	13030	100	8368	100	19069	100
2	3520	97	6447	99	4100	98	9442	99
3	2298	95	4253	98	2678	96	6233	98
4	1693	93	3156	97	1966	94	4628	97
5	1332	91	2498	96	1540	92	3665	96
6	1092	90	2059	95	1255	90	3024	95
7	921	88	1745	94	1052	88	2565	94
8	793	87	1510	93	899	86	2221	93
9	694	86	1327	92	781	84	1954	92
10	615	84	1181	91	686	82	1740	91
11	550	83	1061	90	608	80	1565	90
12	496	82	962	89	544	78	1419	89
13	450	80	877	87	489	76	1296	88
14	411	79	805	86	442	74	1190	87
15	377	78	742	85	402	72	1098	86
16	347	76	687	84	366	70	1018	85
17	321	75	639	83	335	68	947	84
18	298	74	596	82	307	66	884	83
19	276	72	557	81	282	64	828	83
20	258	71	523	80	259	62	777	81
21	241	69	491	79	239	60	731	81
22	226	68	463	78	220	58	690	80
23	211	67	437	77	204	56	652	79
24	199	65	413	76	188	54	617	78
25	187	64	391	75	174	52	585	77
26	176	63	371	74	161	50	555	76
27	166	61	352	73	149	48	528	75
28	157	60	335	72	137	46	502	74
29	148	59	318	71	127	44	478	73
30	139	57	303	70	117	42	456	72

UG13: Track Lengths and Block Sizes

CODE		NAME	RESPONSE TO	CAUSE, RESULT, AND CORRECTIVE ACTION
	0	Normal	Anything	Success
	1*	Imparsible command	Anything	Parse of command failed.
	2*	Restricted command	Anything	Operation, parameter, or parameter value requires different level authorization
PARAMETERS	3	*DSNAME* error	Command	*DSN* missing or bad name.
	4	*BLKSIZE* error	*S I*	*BLKSIZE* missing or too large.
	5	*LRECL* error	*S*	*FB*, *FBS*, *VB* need *LRECL*; or *LRECL*>*BLKSIZE*.
	6	*DISP* error	*SR I*	*DISP=NEW*
	7	*RECFM* error	*I*	*RECFM≠F* (In *IRW*, *RECFM* is determined by the data set).
	8	*UNIT* error	Command	*CATALOG* needs *UNIT*, or data set not cataloged, or no such *UNIT*.
	9	*VOL=* required	Command	The combination *DSN*, *UNIT*, and *DISP≠NEW* needs *VOLUME*.
	10	*NEWNAME* error	*REN*	*NEWNAME* missing or *≠DSN* for PDS.
	11	Duplicate name	*DISP=NEW*	*DISP=NEW* and name already used.
	12	Dataset not found	Command	*DISP≠NEW* and name not found.
	13	Member not found	*S D REN*	Member of a PDS not found.
RESOURCES	15	Data set in use	Command	Data set in use with *DISP≠SHR*.
	16	Volume full	*SW*	Insufficient space for primary allocation.
	17	*PDS* directory full	*SW*	Attempting to write a new member when directory of a PDS is full.
	18	Offline or archived	Command	DASD volume not mounted or request to allocate unit record device denied.
	19	Already cataloged	*SW REN*	(with *DISP=NEW*) Name already cataloged or name conflict as in *A.B.C* and *A.B*
	20	Control variable in use	Command	Last used with *DISP* of *LEAVE* or *REREAD*. Use new variable or retract and reshare.

LEGEND AND NOTES

D DELETE *I IR* or *IRW* *S SR* or *SW* *REN RENAME*	PDS Partitioned Data Set DASD Direct Access Storage	* These codes have a second element that shows the point of difficulty in the command.

UG14: Return Codes -- Response in Command Mode

CODE	NAME	DURING	CAUSE ACTION (None unless stated)
0	Normal	*S I*	Success.
14	Empty vector	*SR*	Empty vector transmitted was data, not end of file. (This is a warning response).
21	Data type error	*S I*	Data type not appropriate to *CODE*. *SR*: Data set closed; to command mode.
22	Data length error	*S I*	*RECFM=F*, *FB*, *FBS*, and *RL≠* (for *CODE≠A*) or < (for *CODE=A*) *BLKSIZE* or *LRECL*, respectively, or *RL>BLKSIZE*. *SR*: Data set closed; to command mode.
23	Data rank error	*W*	*CODE≠A* and data not a vector.
24	File index error	*I*	Improper index (not in ¯1+⍳*N*, where *N* is number of records in data set).
25	*CTL* domain error	*I IR*	*CTL* not a 2-element non-negative integer vector with leading 0 1 2 3 4 or 5, for *I*, or *CTL* did not have leading 0 for *IR*.
26	*DAT* variable required	*I*	*DAT* variable (with suffix = *CTL* suffix) not shared on first data transfer attempt.
27	Variable too large for shared mem.	*R*	Shared variable storage area too small. *SR*: Data set closed; to command mode. (perhaps *CODE=A* assumed erroneously)
28	I/O error	*S I*	Physical error in data transfer. *SR*, *SW*: Data set closed; to command mode *IRW* (writing): Record may have been destroyed.
29	Data set full	*S W*	16 extents have been allocated and filled, or no space on volume for a needed secondary allocation, or primary allocation filled and no secondary allocation specified. Data set closed; to command mode.
30	System error	*S I*	See legend for ⋆. All relevant terminal output should be given to *APL* System Support (see page iv).

CTL Control Variable
DAT Data Variable
RL Record length
S *SR* or *SW*
I *IR* or *IRW*
R Reading

⋆ Second element of response code (for code 30) has the following significance:

1. VTOC full
2. Allocation failed
3. DD card missing
4. System queue error
5. System queue full
6. Directory error
7. *CATALOG* failed
8. *OPEN* failed, due to RACF or hardware error
9. Error in closing or de-allocation

(Also see APLSV Operations Guide, IBM Publication SH20-9088)

UG15: Return Codes -- Response in Data Transfer Mode

Device	2314	3330/3330-II	3340	3344	3350
Bytes/track (unit size)	7294	13,030	8368	8368	19,069
Tracks/Cylinder	20	19	12	12	30
Bytes/Cylinder	145,880	247,570	100,416	100,416	572,070
Cylinders/Volume	200	404/808	348/696	2784	555
Bytes/Volume (millions)	29	100/200	35/70	280	317

UG12: Disk Storage Device Capacities

Code	Key	Use	Effect	Action if in use
2	n	Exclusive use	Immediate	Return code 15
3	n	Exclusive use	When available	Delay until available
4	n	Shared use	Immediate	Return code 15
5	n	Shared use	When available	Delay until available
	↑ Any integer from 1 through ¯1+2*31 may be used. The semaphore is released by assigning a key of 0 with any of the semaphore codes, or by assigning a new semaphore.			

Semaphore Operation Codes

Call for papers: APL81

An international *APL* conference is being planned for next year by the Association for Computing Machinery (ACM). "*APL*81" will be held in San Francisco, California, October 21-23, 1981.

Technical papers on all aspects of *APL*, including the following areas, are solicited:

APL applications (all areas)
APL-- the language
APL implementations (large and small systems)
APL in education
APL in business
APL interfaces with other software systems
APL system organization and management
APL compared to other languages

Both abstracts and full papers will be refereed and authors should submit these documents to the program chairman in accordance with the following schedule:

	Date Due	Author Notification
Abstracts	October 1, 1980	December 1, 1980
Full Papers	February 1, 1981	April 15, 1981

Final copies of complete papers must be received by the program chairman by June 15, 1981 for inclusion in the conference proceedings and for presentation at the conference.

Further information may be obtained from:

Ray Polivka (Vice-chairman of STAPL)
HM1 706
Poughkeepsie, NY (IBM Internal Mail)

T/L 253-3216 or 914/463-3216

or

Richard J. Orgass (Program Chairman)
*APL*81
Xerox Corporation
1350 Jefferson Road
Rochester, NY 14623

or

Eugene R. Mannacio (Program General Chairman)
*APL*81
900 North Point Street
San Francisco, CA 94109

Application for a new account on Kingston APLSV

Return to: IBM Corporation, SCD *APL* Administration,
63C002, Kingston, NY 12401 (8-373-1234)

Manager to be billed__________________ Manager's serial______

Div/Dept_____/______ZIP/ Bldg______ Loc________ Tie-line and ext____________

External mailing address (please include ZIP code)

** Registered IBM Confidential data is <u>not</u> supported **

IBM Data Processing services and resources
are to be used for IBM business only

Serial Number	User name	Location (City)	Div/Dept	ZIP/ Bldg	Tie-line and ext.
....	....	....	....	....	....

Is applicant a regular full-time IBM employee? ☐ Yes ☐ No

Bill to this existing I.T.S. problem number ___________

Issue a new problem number? ______ Billing location __________

Kingston, Poughkeepsie, Charlotte, and Raleigh personnel please provide the following information for a new problem number:

Div____ Maj____ Act____ Proj____ Box____ Dept charged____

× .... × ....

Signature of manager to be billed — Date — Signature of Data Processing Services Coordinator

[To list coordinators, ")*LOAD* 1 *FORMS*" and type "*COORDINATOR*"]

The application cannot be processed unless the above signatures have been satisfied.

<u>Note</u>: When we add your new account to the Kingston *APLSV* system, a verification letter will be SENT TO YOUR MANAGER. The letter will contain your new account number and password, sign-on telephone numbers, and general information. IT IS THEN UP TO YOUR MANAGER TO FORWARD THIS INFORMATION TO YOU. These memos are normally mailed within one week from the time that we receive your application.

Need an Administrative form? Take this to a photocopier, and copy both at once.

Change form

Please return this change form to us if any of your billing or mailing information (division, location, manager, *APL* users in your problem number, etc.) is incorrect. We must have current information for all *APL* users.

```
 ____________________________________________________________________
|            **** Do not write inside this block ****                |
|  New Prob #______         TA Info          Old Prob #______        |
|  Action     Date                           Action     Date         |
|  ------     ----                           ------     ----         |
|                                                                    |
|  Letter    ______                          File copy ______        |
|  Directory ______                          Directory ______        |
|  System    ______                                                  |
|__~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~__|
     Return to:   IBM Corporation, SCD APL Administration,
          63C 002, Kingston, NY  12401   (8-373-1234)
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

** Registered IBM Confidential Data is not supported **

IBM Data Processing services and resources are to be used for IBM business only

☐ Issue new sign-on password (don't use this form for new applications)

☐ Change

☐ Delete sign-on (*only these fields are required)

Manager Information:

Manager	Division	Dept	Bldg
Location (city)	Tie-line/extension		
Manager's serial number*	Signature of manager*	Date*	

User Information:

Sign-on number*	User name*	Tie-line/extension		
Division	Dept	Bldg	Location (city)	User's serial number

Accounting Information: (Kingston, Poughkeepsie, Charlotte, Raleigh, and Burlington users only)

Div	Maj	Act	Project	Box	Dept. charged

Billing location (city)	Coordinator's signature

[To list coordinators, ")*LOAD* 1 *FORMS*" and type "*COORDINATOR*"]

C. USER

1. *GENERAL REQUIREMENTS*: Users of IBM's data processing assets are responsible for compliance with security requirements and for compliance with control requirements specified by owners and by suppliers of services.

   Data processing output is owned by the requesting user, unless other arrangements are made. Users who generate passwords are the owners of the passwords. Information classification and control procedures apply in these cases.

   User management must ensure that owners of IBM's classified information or suppliers of services, as appropriate, are notified upon termination of the user's business need for use of the information, services, or facilities.

2. *ACCESS CONTROLS*: The following access control requirements apply to users.

   Verification passwords and user identifiers that serve as verification passwords must not be shared, and they must be:

   - Classified at least IBM Confidential and controlled accordingly, with particular attention to terminal entry, terminal display, and recording.

   - Deactivated in the event of known or suspected compromise, and the supplier of services notified.

   Information access passwords and cipher keys used as information passwords must be:

   - Assigned the classification of the associated information and controlled accordingly, with particular attention to terminal entry, terminal display, and recording.

   - Controlled so that all persons to whom they have been disclosed are known to the owner or administrator.

   - Administered so that the owner or administrator is notified in event of known or suspected compromise of passwords.

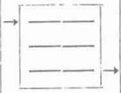
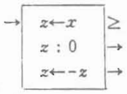
3. *TERMINALS*: Terminal users are responsible for ensuring that:

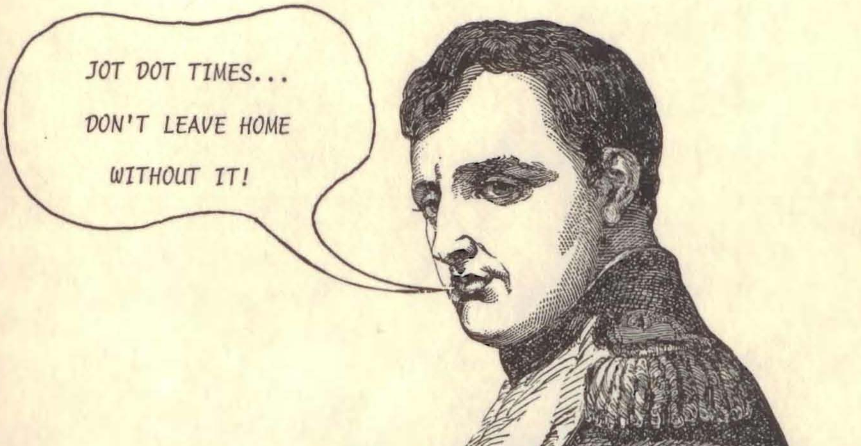
   - Terminals are connected to IBM computers or IBM terminals only for the purpose of conducting internal IBM business and are under IBM control or are in compliance with the control requirements of terminals not under IBM control;
   - Registered IBM Confidential and IBM Confidential-Restricted information are not received by or entered into terminals not under IBM control;
   - Terminals, while unattended, are protected from unauthorized use;
   - Dial terminals, while connected, are attended or otherwise protected;
   - Permanently connected terminals, while logged on, are attended or otherwise protected, and
   - Telephone numbers for computer dial ports are not posted for general view.

* * *

[This instruction replaces Corporate Information Systems Instruction 2-109A, dated January 1, 1978]

記 号 表

関 数	記 号	定 義 か 例 示	参照ページ	計算機用の記号
指 定	$z \leftarrow x$	$z \leftarrow 3$ z に 3 を指定する	5	$Z \leftarrow X$
算 術	$+ - \times \div$		5	$+ - \times \div$
枝分かれ	$\vert x:y \vert \xrightarrow{\mathscr{R}}$	$x \mathscr{R} y$ が真ならば矢印に従って	9	$\rightarrow (X \mathscr{R} Y)/S$
関 係 $\mathscr{R}$	$< \leq = \geq > \neq$	枝分かれと関係関数とに	5	$< \leq = \geq > \neq$
$\boldsymbol{x}$ の成分	x_i	$\boldsymbol{x}$ の i 番目の成分	14	$X[I]$
$\boldsymbol{x}$ の次元	$\rho \boldsymbol{x}$	ρ (3, 4, 5, 6)$\equiv$4	14	ρX
連 鎖	$\boldsymbol{x}, \boldsymbol{y}$	$\boldsymbol{x}, \boldsymbol{y} \equiv x_1, x_2, \cdots, x_{\rho x}, y_1, \cdots, y_{\rho y}$	14	X, Y
関数 F の定義	$z \leftarrow Fx$	$z \leftarrow \vert x$	33	$\nabla Z \leftarrow FX$
				[1] ——— [2] ——— [3] ——— [4] ∇
最 大	$x \lceil y$	$4 \lceil 2 \equiv 4$	34	$X \lceil Y$
最 小	$x \lfloor y$	$4 \lfloor 2 \equiv 2$	37	$X \lfloor Y$
留 数	$m \vert n$	$3 \vert 7 \equiv 1$; $3 \vert {}^{-}7 \equiv 2$; $3 \vert 6 \equiv 0$	37	$X \vert Y$
絶 対 値	$\vert x$	$\vert 3.14 \equiv 3.14$; $\vert {}^{-}3.14 \equiv 3.14$	37	$\vert X$
否 定	$-x$	$-x \equiv 0 - x$	37	$-X$
指 数	$x * n$	$x * 0 \equiv 1$; $x * n \equiv x \times x * n-1$	39	$X * N$
階 乗	$!n$	$!0 \equiv 1$; $!n \equiv n \times !n-1$	38	$!N$
関 係	$x \mathscr{R} y$	$(3 \leq 3) \equiv 1$; $(3 < 3) \equiv 0$	40	$X \mathscr{R} Y$
圧 縮	$\boldsymbol{u}/\boldsymbol{x}$	$(1,0,1,0,1)/\boldsymbol{x} \equiv (x_1, x_3, x_5)$	41	U/X
逆 転	$\Phi\, \boldsymbol{x}$	$\Phi\, 1,2,3,4 \equiv 4,3,2,1$	41	ΦX
整数ベクトル	ιn	$\iota 4 \equiv 1,2,3,4$	41	ιN
低 減	$F/\boldsymbol{x}$	$F/\boldsymbol{x} \equiv x_1 F x_2 F x_3 \cdots F x_{\rho x}$	17	F/X
行列の i 行	M^i	$M^2 \equiv 4,5,6$	66	$M[I;]$
行列の j 列	M_j	$M_2 \equiv 2,5,8,11$	66	$M[;I]$
行列の素子	M^i_j	$M^2_3 \equiv 6$	67	$M[I;J]$
再 構 成	$d \rho x$	$(4,3) \rho \iota 12 \equiv M$; $M \equiv \begin{array}{ccc} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{array}$	69	$D \rho X$
		$12 \rho M \equiv \iota 12$		
多 項 式	$c \prod x$	$c_1 + (c_2 \times x) + (c_3 \times x^2) + \cdots$	54	
自然指数	$* x$	$\left(1, 1, \frac{1}{!2}, \frac{1}{!3}, \frac{1}{!4}, \cdots\right) \prod x$	163	$* X$
双曲線cos	$A x$	$\left(1, 0, \frac{1}{!2}, 0, \frac{1}{!4}, \cdots\right) \prod x$	156	
双曲線sin	$B x$	$\left(0, 1, 0, \frac{1}{!3}, 0, \cdots\right) \prod x$	156	
cosine	$C x$	$\left(1, 0, \frac{{}^{-}1}{!2}, 0, \frac{1}{!4}, \cdots\right) \prod x$	116	
sine	$S x$	$\left(0, 1, 0, \frac{{}^{-}1}{!3}, 0, \frac{1}{!5}, \cdots\right) \prod x$	116	
tangent	$T x$	$(Sx) \div Cx$	126	
双曲線tan	$U x$	$(Bx) \div Ax$	156	
自然対数の底	e	$2.71828 \cdots \equiv 1 + \frac{1}{!2} + \frac{1}{!3} + \cdots$	162	
円 周 率	π	3.14159⋯	12	


JOT DOT TIMES...
DON'T LEAVE HOME
WITHOUT IT!
Jot Dot Times

O . X