ADDITIONS AND CORRECTIONS JUNE 1981

SHARP APL REFERENCE MANUAL

First Edition

March 1979

P. iii, col. 2, last paragraph: Change the sentence to read:

> If you are enrolled in the MAILBOX facility of the I.P. Sharp Toronto APL service, you may, if you prefer, use that means to send your comments to the mailbox code *REF*.

P. x, col. 1, Ch. 12: Change "Limit on total of file reservations" to "File quota and file reservation limit."

P. 2, col. 1, para. 1: The reference to a plane within an array using *A*⌹*B* as an example is incorrect. The sentence beginning "This may..." should be changed to read:

> This may be to each element independently (for example, *A*+*B*); along a particular axis (for example, +/*A*, sum over the last axis of *A*); to a column of *A* in an expression such as *A*⌹*B*; or to an entire array.

P. 3, col. 2, para. 1, addition: The system's **crash recovery facility** was inadvertently omitted. The items now listed as **a b c d** should become items **b c d e** respectively, and the following item **a** should be inserted:

> **a.** Recover from a system crash. Following a crash of the SHARP APL system, every task's active workspace is recovered, either as it was at the moment the system crashed, or as it was not more than four seconds earlier.

P. 10, col. 1, para. 1, addition: To clarify how the timestamp facilitates making "back-up" copies, alter the third sentence to read:

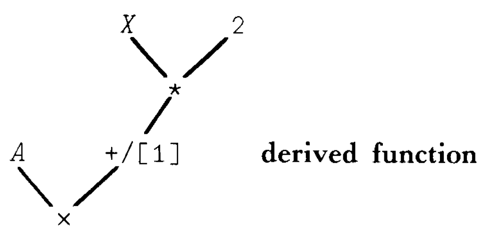
> (The system utilities make "back-up" copies of all active files on a daily basis, as a safeguard against losing data by equipment failure or program error. The timestamp permits the system utilities to detect which components have been changed or added since the last back-up, and to skip those that remain unchanged.)

P. 14, col. 1, para. 1 and small table: Delete these. The symbols discussed are now consistently treated as operators, and the discussion is unnecessary. Were the paragraph to be retained, the reference to "compression" should now be to "replication."

P. 14, "Function" column in outlined table: The first position in this column should read "function symbol or ⎕*NAME*."

P. 19, col. 2, Tree diagram of `A×+/[1] X⋆2`**:** The *X* is misplaced in the diagram. It should appear above and to the left of the ⋆ .

Tree diagram of `A×+/[1] X⋆2`



P. 22, col. 1, paras. 5, 6: Replace these paragraphs with the following:

> **A monadic operator** takes as its left argument whatever is immediately to the left of it. That may be either an array or a primitive function. At present there are two monadic operators, denoted by the symbols / and \ . The effect of these operators depends on whether the argument is a function or an array:

Argument	/	\
Function	Reduce	Scan
Array	Replicate	Expand

P. 22, col. 2, para. 2: The example of an operator sequence is missing a × sign. It should read

```
A×+/[1] X⋆2
```

P. 23, col. 1, para. 6: Change the second sentence so that it says:

> If the function is the argument of an operator, the interpreter evaluates the operator at once (so that what it has now is the derived function that the operator produces).

P. 24, col. 1, para. 3: Change the sentence beginning, "By contrast, in APL..." to read:

> By contrast, in APL the equal sign = is used to denote a proposition (that is, a statement which may be true or false).

P. 24, col. 2, para. 1: In the last sentence introducing the APL example, "if" should be "of" so that it reads:

> But you might concoct an example such as the following, and wonder what the value of *X* should be:

P. 27, col. 1, addition: In the paragraph about branch statements, insert " → Abort execution" so that the example reads:

```
→Y   Branch to Y
→    Abort execution
```

P. 35, col. 1, addition: Insert the following copy before the last paragraph:

> A function whose header shows it to be monadic must be used with a right argument, and may never take a left argument. However, in SHARP APL, a function whose header shows it to be dyadic may nevertheless be used either dyadically or monadically. When a "dyadic" function is used in a monadic sense, it must still have a right argument, but its left argument is omitted. During execution, the name by which the header refers to the left argument has no value.

P. 41, col. 2, para. 4: Replace the present paragraph by the following (to reflect system changes introduced 3 May 1980):

> When the system puts the symbol ⍎ or ⎕ on the state indicator, the corresponding element of ⎕*LC* contains the line number of the function which contains the ⍎ or ⎕. When you enter ⍎ or ⎕ from the keyboard (in immediate execution), the corresponding element of ⎕*LC* contains the line number of the most recently suspended function. However, when there is no defined function on the state indicator, ⎕*LC* contains 0.

P. 42, col. 1, para. 2: Delete the paragraph headed **Watch out**. It is no longer true (because of revisions to the system which became effective 3 May 1980). Also delete a similar warning on page 180.

P. 46, col. 2, para. 2: The section referred to is "Shape of Result and Conformability of Arguments," and not "The Concept of Conformability."

P. 48, col. 1, para. 4: Replace the paragraph labelled "Compress" with the following (reflecting the extended definition adopted 8 February 1980):

> **Replicate:** Denoted by dyadic use of the symbols / or ⌿ . The result of the expression *X*/*Y* has the same shape as *Y* except that along one axis, each position is replicated as many times as are indicated by the corresponding value in *X*. In particular, where *X* contains a 0, that position is not represented at all in the result. When *X* is Boolean, the effect is to compress *Y* by omitting from the result those positions where *X* is 0 and retaining those where *X* is 1.

P. 55, col. 1, para. 2: When characters are used as the argument of ⎕*ARBIN* or ⎕*ARBOUT*, they must be from the first 128 elements of ⎕*AV*. The second sentence should therefore read:

> The argument of ⎕*ARBIN* or ⎕*ARBOUT* is composed of numbers (or of characters from 128↑⎕*AV* to represent numbers in the range 0-127).

The following sentence discusses the result of ⎕*ARBOUT*. As presently worded, it simply says "The result," which might be read as applying to ⎕*ARBOUT* as well as to ⎕*ARBIN*. Hence, the sentence should be amended to read:

> The result of ⎕*ARBIN* is numeric.

P. 58, addition: The character ! can be formed by overstriking . and ' .

P. 62, col. 2, paras. 1-3: The paragraphs numbered **2**, **3**, and **4** should be deleted, and the following inserted:

> **2.** Following)*CLEAR*.
>
> **3.** Following an untrapped error or interrupt. That includes any error in the program itself, or any interrupt signalled from outside it.

4. When execution is complete. That includes completion of a line you entered from the keyboard during execution mode, or following any system command except *)LOAD*. (Following successful execution of a *)LOAD* command, the system immediately executes the expression contained in the visible value of the system variable *⎕LX* (latent expression), which may, in turn, invoke one of the data-entry modes.)

P. 63, col. 1, addition: After the paragraph numbered 3, insert the following:

4. The system disregards any entry which, anywhere within it, contains the escape sequence *O* BACKSPACE *U* BACKSPACE *T* (even when the escape sequence occurs within quotes or as part of a comment).

P. 64, col. 2, para. 1: Delete this paragraph. SHARP APL no longer tolerates LINEFEED RETURN or ATTN RETURN as a way of deleting a line of a function.

P. 65, col. 1, para. 1: The APL example incorrectly specifies 4 BACKSPACES and omits an APL blank. It should appear as follows:

```
Z←'THE TOTALIS: '(5 BS) LF
                  @ IS: 'RETURN
@
```

P. 66-69: The examples of line editing during immediate execution show the recalled line without the six leading blanks with which the system prompts for input. These six blanks are now included in the display, and should appear in the examples. Since the examples (as printed) lack the six blanks, the number shown for positioning the cursor is 6 less than you would use now.

P. 66, col. 1, para. 4: The first sentence of this paragraph should read:

Note that you cannot bring back a line of entry which started with a) or a ∇ , nor one beginning with ⍝ , in immediate execution.

P. 67, col. 1, para. 6: Delete this paragraph. It is no longer necessary due to the rule change discussed for page 71, col. 1, paragraphs 3 and 4.

P. 69, col. 1, para. 1: The example should show 12 backspaces not 13.

P. 71, col. 2, paras. 3-4: The paragraph headed **Inserting blanks you didn't intend** and the one following it should be deleted. The warning is made unnecessary by a change to the rules governing input effective June 1981.

The new rule is as follows: A line of input extends to the position of the cursor at the time you press RETURN, or to the rightmost nonblank character, whichever is further to the right.

This rule does not affect arbitrary input, during which each keystroke or signal is explicitly represented in the result.

P. 73, col. 2, para. 3: Delete the present paragraph, and replace it by the following:

The result of ⍞ input is always a vector.

P. 74, col. 1, para. 5: The description of ways to leave *⎕ARBIN* mode is not quite accurate. The paragraph should say:

Leave *⎕ARBIN* mode by pressing RETURN**:** The escape sequence *O* BACKSPACE *U* BACKSPACE *T* is effective when those are the *only* keystrokes you make. If anything else appears in the entry with them, those keystrokes are treated as part of the entry and are represented as components of the result.

P. 78, col. 1, para. 1: The sentence beginning with "For instance, at an IBM terminal..." should be replaced by the following:

For instance, at an IBM terminal, it's your responsibility to transmit explicitly the characters known as "circle-D" and "circle-C." ("Circle-D" indicates the start of a message or transmission, and "circle-C" marks the end of a message or transmission.)

P. 81, col. 1, para. 2: The function *TPRINT* is in workspace 1 *FILEAID* rather than 1 *HSPRINT*.

P. 81, col. 1, para. 4: The word "font" in the last sentence should be "four" so that it reads:

The four types that are generated....

P. 86, col. 2, first paragraph: Delete the portion of the paragraph at the top of the page, and replace it with the following:

partner. After the degree of coupling becomes 2, a value of the shared variable that is set by either partner becomes visible to both partners. (If, after sharing commences, neither sets a new value for the shared variable, the value that they both see is governed by rules stated in the section headed "Initial Value of the Shared Variable.")

P. 86, col. 2, last paragraph: The paragraph describing functions which do *not* constitute a use of a shared variable should apply only to monadic uses of ⎕*PACK*. The paragraph should read as follows:

4. There are three functions which (while in some sense they "use" the value of a shared variable) do *not* count as a use for the purposes of access control. These are functions whose argument is the *name* of the shared variable (as character data) rather than its value. The three functions are monadic ⎕*PACK* (which incorporates the name and the associated value of a variable into a package), 6 ⎕*WS* (which reports the current value of a variable), and 4 ⎕*WS* (which reports the number of bytes occupied by the current value of a variable).

P. 88, col. 2, para. 4: The statement describing when the clone ID is frozen is ambiguous. The paragraph should read as follows:

Freezing the clone ID: As long as you continue to have any offer to share with another processor (regardless of whether the processor to which you make an offer has made a matching counter offer), your clone ID is said to be "frozen." You can't set it to anything else. When you reduce to zero the number of variables you have offered to share (including any that may at the moment be shadowed), you are again free to change your clone ID. (During the execution of a function containing a local name, the local referent of the name is said to be "visible," and is said to "shadow" other uses of the name.) Offering to share a variable without first setting a clone ID has the effect of freezing your clone ID at the default value 0.

P. 95, col. 1, para. 2: The error message displayed when you interrupt a shared-variable interlock is not *INTERRUPT* but *SV INTERRUPT*.

P. 95, col. 2, para. 1: The reference to ⎕*PACK* should specify *monadic* ⎕*PACK* (for the same reason as noted in the correction to p. 86).

P. 96, col. 1, last paragraph: The description of state change omits some possible cases. The paragraph should read:

State change: A state change is an action (or several actions) outside your workspace but affecting your use of shared variables. "State change" is thus a rather general category. It includes any new offer to share extended to you; any set of a variable you are already sharing, or any set of its shared-variable control; any change in the degree of coupling of a variable you have already offered; or any first use by a partner of a value that you have set (that is, a use which changes the result of ⎕*SVS* for that variable).

P. 97, Functions and text in the box: The four functions and the one line of text should be replaced by the following:

```
        ∇  MONITOR CLONE
[1]        ⎕SIGNAL(CLONE≠⎕SVN CLONE)/555
[2]        SHVARS←0 3⍴USAGE←0 4⍴IDS←0 2⍴0
[3]        INUSE←(1↑⍴NAMES)⍴0                ⍝POSSIBLE INTERNAL NAMES
[4] LOOP:  WORK(⎕SVS SHVARS)[;2]             ⍝PROCESS VARS THAT HAVE BEEN SET
[5]        OLDOFFERS 2=⎕SVO SHVARS           ⍝NOTE RETRACTIONS
[6]        NEWOFFERS ⎕SVQ⍳0                  ⍝ACCEPT NEW OFFERS; <WORK> ON THEM
[7]       →(0∊⍴SHVARS)/0                     ⍝QUIT IF NO ONE TO SERVE
[8]       →⎕SC/LOOP                          ⍝WAIT FOR SOMETHING TO HAPPEN
[9]        ⎕SIGNAL 556                       ⍝INVALID CLONE ID
        ∇
```

The key subroutines used by monitor are:

```
        ∇  WORK SET; SHVAR; USER; N; CH; I; ⎕TRAP
[1]       →(SET∧.=0)/I←0                     ⍝BOOLEAN: WHICH VARS HAVE BEEN SET
[2]        ⎕TRAP←'∩ 6 C →TEST'               ⍝IN CASE HAD NO VALUE WHEN OFFERED
[3]        N←⍴CH←SET/⍳⍴SET                   ⍝CH: NAMES WHOSE VALUES CHANGED
[4] TEST: →(N<I←I+1)↑0
[5]        SHVAR←SHVARS[CH[I];]              ⍝PICK ONE SHARED VAR
[6]        USER←IDS[CH[I];]                  ⍝AND ITS OWNER
[7]        ⍎SHVAR,'←USER  PROCESS ',SHVAR    ⍝USE AND RESET SHVAR
[8]       →TEST
        ∇
```

```
        ∇  OLDOFFERS OK; J; NG
[1]       →(∧/OK)/0
[2]        NG←(~OK)⌿SHVARS
[3]        J←⎕EX NG                          ⍝EXPUNGE ALSO RETRACTS
[4]        SHVARS←OK⌿SHVARS                  ⍝PURGE SHVARS
[5]        IDS←OK⌿IDS                        ⍝PURGE IDS
[6]        INUSE←∨/NAMES∧.=⍉SHVARS           ⍝PURGE INUSE
        ∇
```

```
        ∇  NEWOFFERS X; J; ID; NEW; SURROG; I; N
[1]       →(0=N←⍴X)/I←0                      ⍝X HAS IDS FROM ⎕SVQ ⍳0
[2] TEST: →(N<I←I+1)/END
[3]        ID←X[I;]
[4]        RN←1↑⍴SURROG←⎕SVQ ID              ⍝NUMBER OF NAMES OFFERED
[5]        NEW←RN NAMEFN~INUSE               ⍝UPDATES <INUSE>
[6]        ID←((1↑⍴NEW),2)⍴ID
[7]        J←ID ⎕SVO NEW,' ',SURROG
[8]        J← 0 0 1 1 ⎕SVC NEW               ⍝USER FALLS THROUGH IF I DIE
[9]        SHVARS←SHVARS,[⎕IO] NEW           ⍝UPDATE <SHVARS>
[10]       IDS←IDS,[⎕IO] ID                  ⍝UPDATE <IDS>
[11]      →TEST
[12] END:  WORK(-1↑⍴SHVARS)↑(1↑⍴NEW)⍴1       ⍝IN CASE USER SET BEFORE OFFERING
         ∇
```

P. 98, col. 1, para. 2: Substitute the following definition for the function *NAMEFN*:

```
     ∇  Z←N NAMEFN FREE; LOC
[1]     LOC←N⍴FREE/⍳1↑⍴NAMES
[2]     INUSE[LOC]←1
[3]     Z←NAMES[LOC;]
     ∇
```

P. 100, col. 2, para. 1: Replace the sentence beginning, "For example, if you..." and the one following it with:

For example, if you replace the array stored in one component with another array that takes more space, the new array will (usually) be stored in a new physical location. The space used by the former value is unused (at least for a while). It may be some time until the system is able to reuse the vacated space. Until then, the file will occupy more space than it would if you'd written all of the components in sequence.

P. 101, col. 1, para. 3: Limit on total of file reservations. Replace the paragraph and its heading by the following:

File quota and file reservation limit

The system managers set a limit on the total number of distinct files that you may have (called your "file quota") and on the sum of the outstanding file reservations that you may make (called your "file reservation limit"). These limits apply to all the files you own, regardless of the library in which they're stored, and regardless of whether you actually use the space reserved. When you have created as many files as your quota permits, you will be unable to create another. When you have reached your file reservation limit, you will be unable to increase the limit of an existing file, or to create a new file with a reservation greater than zero. To increase your file quota or your file reservation limit, contact the system Operator or your SHARP APL representative.

P. 101, col. 2, para. 8: Delete the reference to session variables (which is mistaken, as well as irrelevant). The paragraph should then read:

Watch out: A task started by splitting the active workspace appears in many ways identical to the active workspace of its parent. However, it does *not* inherit the parent's file ties, nor the parent's shared variables.

P. 102, col. 1, para. 11: Following the paragraph that begins "If the permission code is ¯1" append the following sentence:

You can also *exclude* particular permissions by subtracting them from a prior permission (for example, from ¯1).

P. 102, col. 2, para. 5: "password" should be "passnumber."

P. 104, col. 1, para. 1: The paragraph discussing what happens when the access matrix is changed is inconsistent with col. 2, para. 2. The discussion is improved by the following changes: (a) delete col. 2, para. 2 (even though there's nothing wrong with it); (b) replace col. 1, para. 1 by the following:

The system selects from the access matrix whatever permission code corresponds to the combination of account number and passnumber that you provided when you tied the file. As long as you keep the file tied, at each file access you must continue to use the passnumber that you supplied when you tied it.

Suppose, while you still have the file tied, the file's access matrix is changed (by you or by somebody else with appropriate access). What happens then? Your authorization changes immediately to whatever permission is now associated with that same combination of account number and passnumber with which you started. If that combination no longer appears in the access matrix, there's nothing left that you can do with the file but untie it (and perhaps tie it again with a different passnumber). If that combination of account number and passnumber is now matched with a new permission code, the new permission takes effect at once.

P. 108, col. 2: Insert the following after item 3:

When the event described in ⎕*ER* occurred during execution of the recovery expression of a ⎕*TRAP*, instead of a reference to the function name and line number, the second row of ⎕*ER* contains the characters ⎕*TRAP* followed by the recovery expression.

An error encountered during execution of a recovery expression is not itself trappable.

P. 110, col. 1, para. 2, addition: A halt to the execution of an N-task or B-task when its limit has been reached is an interrupt. The second paragraph should be amended to read:

A halt to execution caused by the way the STOP control *S*∆ is set is considered an interrupt, and so is a halt to the execution of an N-task or a B-task when the system notes that the task has reached the maximum CPU units or elapsed time that you established when it started.

P. 110, col. 2, addition: Insert the following into the list of trappable errors:

```
25 FILE SYSTEM HARDWARE ERROR
29 FILE LIBRARY NOT AVAILABLE
```

P. 117, col. 1, para. 3: Remove the word *ERROR* following *OPEN QUOTE*. The paragraph should say:

OPEN QUOTE The line contains unbalanced quotes. The message occurs in the context *EDIT OPEN QUOTE* or ⎕*TRAP OPEN QUOTE*.

P. 117, col. 2, para. 2: The last sentence of the paragraph describing use of ⎕*SIGNAL* while debugging is incorrect. It should say:

That aborts execution of the halted function, and brings the function that invoked it to a halt. That is, it makes the next function in the state indicator "suspended" rather than "pendent."

P. 118, col. 2, para. 3: The function *ERRSIGNAL* contains a vector ⎕*TRAP* which lacks a delimiter. Its first line should be:

```
[1]   ⎕TRAP←'∩0 1000 S'
```

P. 121, para. 5: The list of dyadic functions affected by ⎕*CT* should include *X*|*Y*.

P. 122, col. 1, para. 6: The last sentence of this paragraph points out (correctly) that when one of the arguments of a comparison has the value zero, it doesn't matter what value ⎕*CT* has. But the sentence as written doesn't make plain why this is a consequence of the general rules governing ⎕*CT*, and not an exception. The last sentence of that paragraph should be deleted, and the following inserted after the definition of *TNEQ* (col. 2, para. 3):

Because ⎕*CT* can neither be negative nor greater than 1 (indeed, no greater than 16*¯8) it follows that when one argument has the value 0, the other argument can be judged equal to it only when it also is exactly 0, regardless of the value of ⎕*CT*. It doesn't matter what ⎕*CT* is. For example, when *Y* is 0, statement [2] of function *TEQ* reduces to

```
[2]   Z←(|X) XLEQ ⎕CT×|X
```

which can never be true while ⎕*CT* is less than 1.

P. 123, col. 1, addition: Add the following discussion to this chapter:

Tolerant conversion of internal type

Internally, the SHARP APL system represents numbers in one of three ways: Boolean, integer, or floating point. Conversions between these three types are made automatically. In making the conversions, the system is tolerant. That is, it will accept as a reasonable approximation of "Boolean," values which are very close but not exactly equal to 0 or 1. Tolerant conversions and internal data types are discussed in Appendix A. Note that tolerance during type conversion is *not* related to ⎕*CT*.

P. 126, col. 1, last paragraph: The description of roll for an argument 2*31 or greater is defective. Its second line should say:

```
R←⌊⎕IO+Y×((RL+.×2*32 1)÷16*8)÷¯1+2*31
```

P. 130, col. 1, para. 1: The word "fraction" should be "function" so that the sentence reads:

> In APL, residue is also defined for negative left arguments (whereas in conventional arithmetic, the function is usually defined only for a positive modulus).

P. 130, col. 1, para. 3: In the second sentence the *X*'s in the identity should be *Y*'s. This sentence should say:

> But if *X* itself is zero, no number of hops of size 0 will approach any closer to zero, and so 0|*Y* ↔ *Y*.

P. 130, col. 2, para. 1: The residue identity lacks a minus sign and should be corrected to read:

```
(-X)|Y  ↔  -X-X|Y
```

P. 131, table: The first four entries in the row corresponding to ¯0.6|¯3 ¯2.4 ¯1.8 ¯1.2 are shown having the rather surprising result ¯0.6. This was indeed the result returned before the introduction of fuzzy residue in 1978. The system now returns the more reasonable result 0, and those entries should be 0.

P. 134: Insert the following discussion between the sections entitled **The maximum of *X* and *Y*** and **Logical functions** to reflect extensions to the system in 1980.

X∨*Y* **GCD of *X* and *Y*; Logical OR of *X* and *Y***

Arguments: Any pair of conformable numeric arrays. When every element of both arguments is equal to either 0 or 1 (subject to ⎕*CT*), the function ∨ is equivalent to logical OR; see the discussion of "Logical functions" following the sections on GCD and LCM.

Result: A numeric array in which each element is the greatest common divisor (GCD) of each element of *X* and the corresponding element of *Y*. (The GCD function is also known as HCF, for "highest common factor.")

The GCD function can be used to calculate a rational approximation of the value of a fractional array *F*. The representation must show for each element of *F* its sign, an integer numerator, and an integer denominator. This can be done by a procedure such as the following:

```
D←F∨1
SIGN←×F
NUMER←F÷D
DENOM←÷D
```

The greatest common divisor is calculated by the Euclidean algorithm, which can be represented (in direct definition form) as follows:

```
GCD: (α|ω) GCD α : 0=α|ω : |α
```

Note that the sign of a divisor is ambiguous, and so, to insure that the function is both commutative and associative, it is defined always to be positive (or zero).

Since GCD is defined in terms of residue, and residue is subject to comparison tolerance ⎕*CT*, GCD is also subject to ⎕*CT*.

Watch out: Prior to the implementation of GCD in 1980, the symbol ∨ denoted only logical OR, and was unaffected by ⎕*CT*. It is possible that a program written before that time ran correctly even when ⎕*CT* was undefined. Such a program now requires that ⎕*CT* be defined.

X∧*Y* **LCM of *X* and *Y*;**
Logical AND of *X* and *Y*

Arguments: Any pair of conformable numeric arrays. When every element of both arguments is equal to either 0 or 1, the function ∧ is equivalent to logical AND; see the discussion of "Logical functions" following this section.

Result: A numeric array in which each element is the least common multiple (LCM) of each element of *X* and the corresponding element of *Y*.

The LCM is evaluated by the following algorithm:

LCM: α×ω÷α *GCD* ω

Note that the sign of the result is the sign of the product of the arguments.

Watch out: Prior to the implementation of LCM in 1980, the symbol ∧ denoted only logical AND, and was unaffected by ⎕*CT*. It is possible that a program written before that time ran correctly even when ⎕*CT* was undefined. Such a program now requires that ⎕*CT* be defined.

P. 137, col. 2, addition: Insert the following under the figure:

A is the shaded area and also the length of the arc *IP*.

P. 138, addition: The argument *B*, shown in the figure illustrating the hyperbolic functions, is not defined. It is the argument of the hyperbolic function. On a unit circle (where *OP*=1), its value is equal to the area that is shaded in the figure. Insert on the figure:

B is the shaded area;
B has no equivalent arc.

P. 139, col. 1, para. 3: In the description of the point *Q* lying on the unit hyperbola, delete the phrase "with coordinates *U* and *T*." The coordinates of *Q* are *QT* and *OT*.

P. 139, col. 2, last sentence: The sentence confuses the terms "domain" and "range." It should be deleted and replaced by the following:

The domain of the artanh function ¯7○ is numbers whose magnitudes are less than 1.

P. 141, col. 1., para. 1: The last sentence should be changed to read:

If you attempt to use reshape to create a nonempty array from an empty right argument, the system rejects the expression with the message *LENGTH ERROR*.

P. 142, col. 2, para. 3: "Bracket" should be "brackets."

P. 143, col. 2, para. 4: Again, "bracket" should be "brackets."

P. 144, col. 1, para. 2: Within the parentheses there should be no "a" before the 100. The phrase should read, "(or by 20 or by 100, etc.)."

P. 145, col. 2, para. 7: Since the word "dense" has a particular meaning in set theory, the first two sentences should be altered to avoid it, thus:

The elements of *X* must be integers chosen so that (when the desired result has rank *R*) *X* includes each member of ⍳*R* at least once.

P. 145, col. 2, para. 8: The symbol ⍋ is missing from the expression for the shape of the result of dyadic transpose. The order of the identities should be reversed, and the text should say:

Suppose *Z*←*X*⍉*Y*, then

```
      ρY ↔ (ρZ)[X]
      ρZ ↔ (ρY)[⍋X]
```

P. 146, col. 2, para. 3: The expression should be:

```
ρX⍉Y ↔ (ρY)⌊.+(⌈/ρY)×X∘.≠⍳⌈/0,X+~⎕IO
```

P. 153, col. 2, para. 1: The first *Y* in this paragraph should be replaced by *X*, so the sentence begins:

Where an element of *X* is negative,....

P. 153, col. 2, para. 4: In the last sentence, "effect" should be "affect."

P. 154, col. 1, para. 6: The SHARP APL system optimizes the evaluation of expressions in the form

```
(BOOLEAN)/⍳X
```

Both references to ρ now occurring in that paragraph are irrelevant, and should be deleted.

P. 157, col. 1, Discussion of compression: The headings and the first paragraph should be changed so that they read as follows:

X/*Y* *X* **replicating** *Y*
X⌿*Y* *X* **replicating the first axis of** *Y*

Replication alters the length of one of the axes of *Y* according to values in the left argument *X*.

P. 157, col. 1, para. 6: The first sentence describing the left argument should be amended to the following:

Left argument: A numeric vector or scalar each of whose elements is a non-negative integer.

P. 157, col. 1, para. 7: The word "compressed" in the first sentence should be changed to "replicated," so that the sentence says:

The length of *X* must be equal to the length of the axis of *Y* that's being replicated.

P. 157, col. 1, para. 8: "Replication" should be substituted for "compression" in the heading and in the first sentence so that they read as follows:

Axis of replication: Replication always takes place along just one axis.

P. 157, col. 2, para. 2: The first sentence should be altered to read:

In the result, each position on the axis along which replication takes place is replicated as many times as indicated by the value of the corresponding element of *X*. When *X* is Boolean, the result retains those positions from *Y* in which the corresponding element of *X* is 1, and lacks those positions where the corresponding element of *X* is 0.

The remainder of the paragraph should be amended to say:

The various axes of the result have the same lengths as the corresponding axes of *Y*, except for the axis along which replication takes place. The length of that axis is +/*X*, or, when *X* is a scalar, *X* times the length of the axis of *Y*.

P. 159-160: The definitions shown for *INTERSECTION* and *UNION* are incorrect. They should be:

```
      ∇ Z←X INTERSECTION Y
[1]     Z←(X∊Y)/X
      ∇

      ∇ Z←X UNION Y
[1]     Z←X,(~Y∊X)/Y
      ∇
```

P. 163, col. 1, para. 5: In the discussion of **Scalar radix,** the remark that the effect of *X*⊤*Y* is identical to *X*|*Y* neglects to mention that | is subject to comparison tolerance, while ⊤ is not. The last sentence should be:

> When ⎕*CT* is zero, the effect is identical to *X*|*Y*.

P. 164, col. 2, para. 2: The discussion of complete representation incorrectly states that there's no way to completely represent a positive number by using a negative radix. This paragraph should read:

> When all the elements of the radix are positive (and so all column values are positive, too), there's no way you can get a complete representation of a negative value. Instead, you'll get a *complement.* For example:

```
      10 10 10 10 10 ⊤ ¯12
9  9  9  8  8
```

P. 165, col. 1, para. 2: The shape of the result of *X*⊥*Y* is `(¯1↓ρX),1↓ρY` not `(¯1↓ρX),¯1↓ρY)`. Thus, the last sentence should read:

> The shape of the result is `(¯1↓ρX),1↓ρY`.

P. 165, col. 1, para. 3: In the last sentence, "introudced" should, of course, be "introduced."

P. 168, col. 2, para. 3: The discussion of the result of *X*⌹*Y* incorrectly states the criteria. It should say:

When *Y* **is a square matrix:**

```
    Y+.×X⌹Y ↔ X
```

When *Y* **has more rows than columns:**

```
    +/+/(X-Y+.×X⌹Y)*2 is a minimum.
```

P. 173, col. 2, para. 8: There is a mistaken reference to **w** in the description of the effect of a negative value in the left argument of ⍕ . That paragraph should say:

> **Negative:** Exponential format, with |**d** significant digits

P. 174, col. 1, last paragraph: The example of dyadic ⍕ unaccountably shows ⍉*X* rather than *X*. The example should be:

```
      5 0  8 3  5 2 ⍕X
    1   0.500 0.33
    0   0.200 0.17
    0   0.125 0.11
```

P. 174, col. 2, para. 2: There is a mistaken reference to **d** in the discussion of field width zero. That paragraph should start out:

> **Field width** 0 **lets the system pick the width:** When **w** is 0....

P. 175, col. 2, last paragraph: The example shows 4 significant digits rather than 2. The last paragraph of text and the accompanying example should therefore be as follows:

> and in exponential form, with 4 significant digits for each column, like this:

```
      0 ¯4 ⍕ Y (or alternatively ¯4⍕Y)
 2.100E1    1.235E4   ¯1.154E0
 1.700E1    1.235E4    3.679E0
¯6.000E0    1.235E4    2.665E0
```

P. 178, col. 1, para. 1: Although the text just before the example states that the second column is to be represented with 6 digits, the example specifies 5. The text is correct and the example should be changed. Also in the example, the result shows more significant digits than are called for in the argument. (That's because in an earlier version of SHARP APL, the system allowed up to three print positions for the exponent, and when they weren't needed, gratuitously supplied one or two additional digits.) The example should appear as follows:

```
  12 ¯3  12 ¯6  12 5 ⍕ Y
 2.10E1   1.23457E4   ¯1.15444
 1.70E1   1.23457E4    3.67890
¯6.00E0   1.23450E4    2.66500
```

P. 180, col. 1, para. 5: Amend the paragraph so that it reads as follows:

> **Watch out:** When you have control at the keyboard, but there's a suspended function on the state indicator *and* there's an active ⎕*TRAP*, an invalid instruction that you enter from the keyboard may be trapped. If the trap action is *C* (and ⎕*TRAP* is localized at a level earlier than the suspended function), the function's execution may be aborted. If the trap action is *C* or *E* and the recovery expression contains a branch, execution may be automatically resumed.

P. 180, col. 1, para. 6: In the paragraph headed **Effect when destination is zero:** delete the last two sentences beginning with "**However**, if the branch statement...."

P. 180, col. 1, paras. 7-8: Replace these paragraphs with the following text:

> **Effect when destination is empty:** During execution of a defined function, branching to an empty vector has no effect. Execution continues in sequence with the next statement on that line of the function (if there is one), or (if there is no further statement) by a normal exit from the function.

P. 180, col. 2, para. 2: Because of revisions to the system which became effective 3 May 1980, this paragraph should be deleted. (See also the deletion note for page 42, column 1, paragraph 2.)

P. 185, summary in box: The first three references to *Y* should be to *N*. Thus the first two entries should be:

+\N	Cumulative sum of *N*.
×\N	Cumulative product of *N* (for example, over successive rates of growth or attrition).

P. 185, col. 1, last paragraph: The reference to ∧/*Y* should say ∧/*B*.

P. 188, col. 2, para. 2: Change "compress" to "replicate."

P. 189, col. 2, last paragraph: The expression for the shape mistakenly refers to the arguments as *A* and *B*. The correct expression is:

```
SHAPE←(⍴X),⍴Y
```

P. 190, col. 1, para. 2: Change "is" to "was supplied" so that the paragraph reads:

> Of course, the labelling in the foregoing illustration was supplied afterwards; the result of the outer product is simply the array of values that appear as the body of the table.

P. 193, col. 1, para. 2: The first line of the example

```
AXIS←(⍳⍴⍴X)-⎕IO=0
```

should be changed to read:

```
AXIS←(⍴⍴X)-⎕IO=0
```

P. 193, col. 2, para. 1: The successive powers should (of course) be assigned not to *COEFF* but to the variable *POWERS*, so the expression should be:

```
POWERS←X∘.*(⍳1↑⍴COEFF)-⎕IO
```

P. 194, col. 1, para. 1: The example is missing a ⍉ . It should read:

```
IOTA+.×REFERENCE∧.=⍉SAMPLE
```

P. 196, para. 1: Strictly speaking, you can't *start* a B-task; you can only submit a request that the system start one for you. Also, the name of the appropriate workspace is 1 *BTASKREQ*. Thus, the third sentence should begin:

> To submit a B-task request, you use the defined function *BTASKREQ* in workspace 1 *BTASKREQ*....

P. 197, top of 2nd column: It is not true that *⎕HT* is propagated into an N-task. The statement should say:

> ...the visible value of the session variable *⎕SP* is the same value that *⎕SP* had in the parent task's active workspace....

P. 198, col. 2, Argument of *⎕BOUNCE*: The argument of *⎕BOUNCE* can be a scalar or a vector, so the sentence should be changed to read:

> **Argument:** A scalar task number or a vector of task numbers.

✓ **P. 203, col. 1:** In the table of decorators, a phrase that occurs in the descriptions of the other decorators is omitted in the description of decorator *P*. No distinction is intended. To make the description of *P* parallel that of decorator *M*, the table should say:

P<**text**>	Print **text** to the left of the leftmost digit of a non-negative number.

P. 205, col. 1, para. 1: There are five (not four) possible values for the left argument of *⎕FD* (as the table correctly illustrates).

P. 206, col. 2, para. 6: Two points about the use of *⎕FX*: although groups are considered obsolete, while they still exist, you can't create a global object whose name conflicts with one; the restriction on editing a function on the stack refers only to the visible use of a name. So the paragraph should be revised as follows:

> When you're using *⎕FX* to fix a function definition, there must be no existing visible use of the proposed function name as the name of a variable or label. (If the proposed function is global, it can't have the same name as a group.) When the visible referent of the proposed name is a function, you *may* use *⎕FX* to replace its present definition with a new one *provided* it is not being executed (that is, provided it doesn't appear anywhere in the state indicator).

P. 207, col. 2, para. 1: The numeric return code from 3 *⎕FD* is not accurately described. The third sentence of that paragraph should be deleted, and replaced with the following:

> The function 3 *⎕FD* returns a 2-element numeric vector. The first element is a code indicating what sort of problem was encountered. The second element indicates on what line of the proposed function the trouble was found. When the right argument was a matrix, the result indicates the row which contains the defect. When the right argument was a vector, the result indicates the position in the vector at which the defective line begins.

P. 207-208: The discussions of 6 *⎕FD* and 7 *⎕FD* mistakenly refer to the error message *⎕FD ERROR*, which no longer exists. The text should say *DOMAIN ERROR*.

P. 208, col. 1, para. 2: The result of *⎕EX* is always a vector. The paragraph should say:

> **Result of** *⎕EX*: A Boolean vector containing one element for each name to be expunged.

P. 208, col. 1, para. 5: The result of 6 *⎕FD* is in alphabetical order. Thus, the first sentence should read:

> A character matrix containing, in alphabetical order, all those names that could not be expunged.

P. 208, col. 2, para. 2: The result of `7 ⎕FD` is also in alphabetical order. Hence, the paragraph should begin as follows:

> A character matrix containing, in alphabetical order, those names....

P. 218: Permission code `1024` does not include *⎕SIZE* access. In the first table, the entry for `1024` should show *⎕RESIZE* and *⎕STIE* (but not *⎕SIZE*). In the second table, *⎕SIZE* should show permission `1` (but not `1024`).

P. 220, col. 1, para. 2: Discussion of arguments of *⎕HOLD* or *⎕FHOLD* when files are tied using passnumbers. Delete the paragraph starting "If you used..." and substitute the following:

> Suppose that you want to hold several files, and that when you tied them, you tied at least one of them with a passnumber. To hold those files, you may use either *⎕HOLD* or *⎕FHOLD*, depending on the permission granted in their access matrices. If you use *⎕HOLD*, the argument is (as always) a vector of the tie numbers of the files you want to hold. *⎕HOLD* is unaffected by passnumbers. But if you use *⎕FHOLD*, the argument is a 2-row matrix whose first row contains the tie numbers and whose second row contains the corresponding passnumbers (or `0` for each file tied without a passnumber). Because *⎕HOLD* is unaffected by passnumbers, a file that is used for an application that requires passnumbers is usually given an access matrix that prohibits use of *⎕HOLD*.

P. 220, col. 2, para. 5: Discussion of effect of untying a held file. The last sentence of the paragraph numbered **2** mistakenly says "tied" where it should say "held." That sentence should say:

> However, those that you untie are free to be held by another task.

P. 221, col. 1, last paragraph: Both *⎕HOLD* and *⎕FHOLD* appear in the example, but in a context in which there is no difference between them. While the example is not wrong as shown, it may imply some distinction between them. It would therefore be preferable to use *⎕FHOLD* throughout, so that the last paragraph becomes:

> Note also that *⎕HOLD* or *⎕FHOLD* can be used to synchronize activities other than file operations. For example, two tasks which each want to bounce the other might use
>
> ```
> ⎕FHOLD Y
> ⎕BOUNCE 1↓⎕RUNS[;1]
> ⎕FHOLD ⍳0
> ```
>
> where *Y* is some mutually agreed-on file.

P. 223, col. 1, last paragraph: Replace the paragraph to reflect system changes introduced on 3 May 1980:

> The primitives ⎕ and ⍎ appear on the state indicator in the same way as defined functions. Where one of those occurs in the state indicator, the corresponding element of *⎕LC* contains the line number of the statement in the defined function which invoked ⎕ or ⍎ . When ⍎ or ⎕ was invoked from immediate execution, the corresponding element of *⎕LC* contains the line number of the most recently suspended function (if there is one). However, when ⎕ or ⍎ is used from immediate execution and no defined function is on the state indicator, *⎕LC* is `0`.

P. 225, col. 1, para. 1: The second "the" in the second sentence should be removed so that the sentence reads:

> It takes as its left argument one of the integers from 1 to 6.

P. 226, table at top of first column: `4` is not a permissible argument for *⎕NL* (although it is for `1` *⎕WS*), and should be deleted from the left side of the table.

P. 227, discussion of `2` *⎕WS* `4`: In a workspace that has not been saved, the third element is undefined (but it isn't `0`). Thus, point **3** should read:

> **3.** Timestamp showing the date and time at which the workspace was saved (undefined in a workspace that has not been saved)

P. 228, col. 1, para. 2: The reference to a group that has no members is meaningless. The sentence should say:

> If the visible referent of the name contained in *Y* isn't the name of a group, the result is an empty matrix.

P. 229, col. 1, last paragraph: Two characters have been omitted from the APL expression to select the visible nameclass from the result of 5 ⎕*WS*. The correct expression is:

```
VISIBLE←((+/∧\QWS<0)⌽QWS)[;⎕IO]
```

P. 236, col. 1, para. 4; P. 237, col. 2, para. 4: "ouput" should be "output."

P. 237, col. 1, para. 7: The restriction on the length of an input line has been removed. Replace the present paragraph by the following:

> You can make one line of entry. Your entry continues until you press RETURN. There is no fixed limit to the number of characters you can safely type before you press RETURN. You can always enter about 150 characters. If the system at the moment has sufficient space available in its typewriter buffers, and your active workspace has sufficient space to receive the data they represent, you *may* be able to transmit many more characters—even several thousand—but it would be unwise to count on that.

P. 237, col. 2, para. 2-3: Delete the present paragraphs and replace by the following:

> **Result:** A character vector containing the contents of the ⍞-output buffer (which may be empty) overstruck with the entry supplied from the keyboard.

P. 238, col. 1, para. 3: The name of the variable is *R*, not *NAME*, so the last part of the sentence should read:

> ...then the value of *R* will be the 21....

P. 238, col. 1, para. 5: In the example, *PROMT* should be *PROMPT*.

P. 247, col. 1, para. 5, discussion of ⎕*PVAL*: The error message should (of course) be spelled *DOMAIN ERROR*.

P. 247, col. 2: Insert the following as item 4 in the list of criteria for the left argument of ⎕*PDEF* (making the present fourth item number 5):

> **4.** It must not be a name whose visible referent in the workspace is a label (or, while groups still exist, a group).

P. 251, col. 2, para. 6: In the last sentence, "interpet" should, of course, be "interpret."

P. 254, col. 1, para. 1: The first paragraph should be replaced by the following text:

> **Argument:** An integer scalar. When *Y* is positive or zero, it is the proposed clone ID. Its value must not conflict with the clone ID of another workspace running under the same account number.
>
> It is permissible to supply a negative value for *Y*; the effect is to inquire about the current value of the clone ID.

P. 254, col. 1, para. 2: The discussion of conditions that cause the clone ID to be frozen is inaccurate. The final parenthetical sentence in that paragraph should read as follows:

> (The clone ID is frozen as long as you have any offer to share with another workspace, regardless of whether the other processor has made a corresponding offer to you.)

P. 254, col. 1, para. 5: ¯1 should be replaced by "negative," so that the sentence reads:

> Note that when the argument of ⎕*SVN* is negative, the result is the current clone ID.

P. 254, col. 2, para. 1: While SHARP APL still includes groups, the last sentence should be changed to include them:

> Thus, it can't be a label, or the name of a function (or a group).

P. 256, col. 1, para. 6, addition: Insert this addition between the two paragraphs discussing the result of ⎕*SVC*.

> When the argument *Y* is a matrix, the result of ⎕*SVC Y* is a 4-column matrix having the same number of rows as *Y*. When *Y* is a vector or a scalar, the result of ⎕*SVC Y* is a 4-element vector.

P. 256, col. 2, para. 6: Discussion of 0 value for state-change variable. Replace the present paragraph by the following:

> When your active workspace lacks a valid clone ID, each time you use ⎕*SC* the system immediately supplies a clone ID of 0. Your workspace lacks a valid clone ID when the default clone ID of 0 is already in use by another task running on the same account, and you have not used ⎕*SVN* to establish a different clone ID.

P. 256, col. 2, last paragraph, and all text on page 257: Replace the text with the following discussion:

> In general, ⎕*SC* returns a 1 when there has been a state change *since the last time you used* ⎕*SC*. But suppose you haven't used ⎕*SC* before? In that case, ⎕*SC* returns the value 1 when there has been a state change since you *started* using shared variables (that is, since you first used ⎕*SVQ* or dyadic ⎕*SVO*).
>
> Suppose you haven't yet started to use shared variables? There isn't much point in using ⎕*SC* in that case. It's a bit like asking whether there's been a state change ever, in all time (since there is no defined point of beginning). However, if you do use ⎕*SC*, you get back the value 1 immediately.
>
> Once you have used ⎕*SC*, or have begun to use shared variables, the system inhibits you from using ⎕*SC* again until there has been a state change. The value of ⎕*SC*, *when* you receive it, is 1. *But you don't receive that value unless or until at least one state-change event has occurred.*
>
> If (by the time you attempt to use ⎕*SC*) the system has already detected a state change, you receive the value 1 immediately. If no state change has taken place since the last time you used ⎕*SC*, your new use of ⎕*SC* is delayed indefinitely until some state change occurs.
>
> **State-change events:** A state-change event may be any of the following:
>
> **1.** Someone extends to you a new offer to share. An incoming offer is "new" if it does not match any outstanding offer you've already made, and the offeror doesn't already have that offer outstanding to you. You detect new offers with ⎕*SVQ* ⍳0.
>
> **2.** A partner sets the value of a variable you are already sharing. You can usually detect such a set because the result of ⎕*SVS*, for that variable, will have become 0 1 0 1. (However, if the variable's state was *already* 0 1 0 1, you won't be able to tell what your partner did.)
>
> **3.** A partner sets ⎕*SVC* for a variable you are already sharing. You may detect this by comparing the result of ⎕*SVC* for all variables with its earlier value (provided you stored that earlier value). But if your partner set ⎕*SVC* to the *same* value it had before, you won't be able to tell what happened.
>
> **4.** There's a change in the degree of coupling of a variable you have previously offered to share. This might be a change from 1 to 2 (when a partner accepts an offer of yours), or from 2 to 1 (when a partner retracts the offer to share a variable for which sharing is already established).
>
> You can detect either of these changes by comparing the present and prior result of ⎕*SVO* for the names you've offered.

5. A partner uses for the first time a value that you have set. You can detect such a change because the result of ⎕*SVS*, for that variable, becomes `0 0 1 1` when before it was `1 0 1 0`.

Watch out: When you receive the value `1` from ⎕*SC*, you know only that *at least* one of those events has occurred. You don't know how many events occurred, or what they were. It is therefore essential to check all those events that are possible and which may be significant in your application. Some of these checks require that you maintain a record of the prior state of affairs (such as the names you have offered, their degree of coupling, or their state) so that you can compare the current values with those from the last time you used ⎕*SC*.

There are some events which count as state changes but which you may be unable to detect. That would arise if the variable involved is shadowed. It would also arise if your partner reset a variable you had not yet used, or reset ⎕*SVC* in a way that produced no effective change. Presumably such hidden events arise from errors in the design of the programs that are attempting to cooperate.

P. 259-262: For each of the system variables discussed in the chapter (other than ⎕*LX*), only a scalar value is meaningful. The phrase "or a 1-element vector" should be deleted from the discussion of ⎕*CT* (p. 259-col. 2, para. 4), ⎕*IO* (p. 260-col. 2, last para.), ⎕*RL* (p. 261-col. 2, para. 7), ⎕*PP* (p. 262-col. 1, para. 1), and ⎕*PW* (p. 262-col. 2, para. 5).

P. 259, col. 2, para. 4: The value of `16*¯8` is incorrectly shown as a negative number. The sentence should read:

The system requires that ⎕*CT* be a non-negative scalar less than `16*¯8` (which is about `2.328306437E¯10`).

P. 265, col. 2, para. 4, addition: Change the heading for the description of)*BLOT* so that it reads:

)*BLOT* **[n]** **Print a blot**

Insert the following text between the two paragraphs describing)*BLOT*:

When you provide an argument **n**, the blot starts at the left margin and extends for **n** print positions. **n** must be an integer between `0` and ⎕*PW* (inclusive).

P. 268, col. 1, para. 9: While alphabetizing displays of names, the system treats the character ∆ as if it follows *Z*. The digits `0-9` follow all alphabetic characters. Hence the sequence of characters should be:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ∆
A̲B̲C̲D̲E̲F̲G̲H̲I̲J̲K̲L̲M̲N̲O̲P̲Q̲R̲S̲T̲U̲V̲W̲X̲Y̲Z̲∆̲
0123456789
```

P. 268, col. 2, last paragraph: The system command)*LIB* has been extended, so the heading should be changed to the following:

)*LIB* **[libno] [letters]** **List workspaces in library**

P. 269, col. 1, para. 3: Replace this paragraph by the following discussion which reflects the extensions to)*LIB*.

The system displays a list of the workspaces in the library you've requested, horizontally and in alphabetical order. If you include one or more letters to the right of the)*LIB* command, the system begins its display at that point in the alphabetical sequence of workspace names. As with the commands)*FNS* and)*VARS*, the letters you use for this purpose don't have be present in the names of workspaces in the library.

P. 269, col. 2, para. 7: To indicate that a number is locked out, the system sends the message *NUMBER LOCKED OUT OF SHARP APL SYSTEM*, (not *LOCKED OUT*).

P. 273, col. 2, last paragraph: Correct the first sentence to read:

> **2.** No locked function in a sealed workspace can be erased.

P. 275, col. 2, para. 5: To reflect changes introduced 5 April 1980, replace the paragraph starting "The principal types..." with the following:

> When you first sign on, the system transmits data in a way suited to simple hardcopy terminals, including a calculated number of idles. The system deduces from the characteristics of your signon whether the character set is ASCII, or one of those apppropriate to the IBM 2741 or the Teletype 33.
>
> To suppress the idle characters, enter

```
        )TERMINAL NOIDLES
```

> The system responds by displaying the previous state:

```
WAS IDLES
```

> You can restore the transmission of idles by typing:

```
        )TERMINAL IDLES
WAS NOIDLES
```

> The setting *)TERMINAL NOIDLES* may give you a significant increase in the speed of data display, as well as a decrease in the number of characters transmitted by the system. However, if you use this setting at a buffered hardcopy terminal, it may begin to lose characters and behave erratically if its internal buffer becomes completely filled.
>
> The system also accepts as an argument to *)TERM* a keyword indicating one of certain specific terminal types, such as *GET*1200, *SDATA*, *TY*33, and so on. An up-to-date list showing what terminal types are supported can be found by typing *TERMINALS* in workspace 5 *TERM*.

P. 279, col. 2, para. 2: The message *CHAR ERROR* now refers only to an unacceptable character within an array being interpreted (for example, within a *⎕TRAP* recovery expression). Illegal overstrikes during keyboard entry are now rejected with the message *ENTRY ERROR*, which thus replaces *CHAR ERROR* in that context.

P. 280, col. 1, additions: Insert two additional error messages, as follows:

> *EDIT OPEN QUOTE*
>
> Your entry line contains an opening quote without a matching closing quote (that is, an odd number of quotes). The system displays the line as entered, and then positions the cursor on the next line at about the middle of the line you typed. The system automatically switches to editing mode, and awaits an instruction that will edit the line just displayed. You may use slashes to indicate deletions, and a dot or a comma to mark the position at which you wish to insert additional characters. See the description of line editing in Chapter 10.
>
> **Watch out:** After you've received the message *EDIT OPEN QUOTE*, the system interprets whatever you type next as an effort to edit the line is has just displayed, and *not* as a new entry. You can abort line editing by entering the escape sequence *O* BACKSPACE *U* BACKSPACE *T*.
>
> *ENTRY ERROR*
>
> The input processor has detected an overstrike that does not form a valid APL character. It echoes to you the characters it has received up to the invalid character, and waits for you to retransmit the remainder. Note that this error is detected before your entry reaches the APL interpreter, and cannot be trapped.

P. 281, col. 1, addition: Insert the message and text which follows between the discussions of *FILE INTERRUPT* and *FILE NAME ERROR*.

> *FILE LIBRARY NOT AVAILABLE*

The system's physical storage of files is divided into segments called "volumes." The volume in which a file is stored depends on the number of the library to which it belongs. The volume containing the file you have requested is at present unavailable. That's usually because of maintenance, or perhaps because of trouble with the physical device on which that volume is stored. The problem may be transitory, but if it persists the system Operator may be able to advise you when the volume (and hence that file library) will be available. (Trappable as event 29.)

P. 281, col. 2, addition: Insert the following message and text between the discussions of *FILE SYSTEM ERROR* and *FILE SYSTEM NO SPACE*.

FILE SYSTEM HARDWARE ERROR

The system is unable to read correctly from the physical device used to store the component you've asked to read or write. Report the occurrence to the system Operator so that corrective action can be taken. (Trappable as event 25.)

P. 281, col. 2, para. 4: Replace the paragraph headed *FILE SYSTEM TIE CAPACITY EXHAUSTED* with the following:

FILE SYSTEM TIES USED UP

The system now has tied the maximum number of distinct files. (The number depends on the installation.) The situation may be temporary. You should bring the occurrence to the attention of the system Operator so that steps can be taken to decrease the likelihood of a recurrence. (Trappable as event 30.)

P. 283, col. 2, addition: Insert the following discussion between the descriptions of *LIBRARY TABLE FULL* and *LINES DOWN*.

LINE TOO LONG

The system line editor cannot handle a line of more than 250 characters. You have attempted to edit such a line, or, while editing, have indicated changes that would make the line exceed that length.

P. 283, col. 2, para. 3: The paragraph under *MESSAGE LOST* should read:

The message you entered with the command *)MSG*, *)MSGN*, *)OPR*, or *)OPRN* has not been sent because the system received an attention signal from your terminal after you sent the message but before the system dispatched it.

P. 284, col. 1, para. 1: The first sentence should be changed to read:

The listed items have not been copied either because their names conflict with protected names in your active workspace, or because you've tried to copy a locked function from a sealed workspace.

P. 284, col. 1, para. 2: Insert the word "locked" in the second sentence so that it says:

An object is not erased if it's a locked function in a sealed workspace, or an active function.

P. 284, col. 2, addition: Insert this description of *OPEN QUOTE* between the discussions of *NUMBER NOT IN SHARP APL SYSTEM* and *PATIENCE PLEASE*.

OPEN QUOTE

Your entry line contains an opening quote without a matching quote to close it (that is, an odd number of quotes). The system displays the line, and waits with the cursor at the end of it. You may supply a closing quote (if that's what is needed) or you may backspace to the error, press LINEFEED or ATTN to indicate a correction, and then enter the balance of the line.

P. 285, col. 1, addition: Insert the following description of *RESEND* between the discussions of *RANK ERROR* and *SENT*.

> *RESEND*
>
> The system has received an undecipherable entry from your terminal, and is unable to salvage any of it. The only remedy is for you to transmit your entry again. (Note: *RESEND* is a message you are unlikely to encounter while working through the Sharp network. That's because automatic error checking between the various minicomputers is usually able to correct the problem before your message reaches the central computer's transmission control unit.) A *RESEND* message can also arise when you send an extremely long entry without using the RETURN key. The exact permissible length may vary from moment to moment with the availability of input buffers at the central computer. Entries of excessive length rarely arise while you're keying your entry manually from the terminal. They are more likely when you play a recording of entries generated elsewhere to simulate entry from a terminal.

P. 286, col. 2, addition: Insert the following description of *SYSTEM FULL* between the discussions of *SYSTEM ERROR*, *CLEAR WS* and *SYSTEM RESET DETECTED*.

> *SYSTEM FULL*
>
> You have successfully connected to the local dial access, and it in turn is in contact with the central computer's transmission control unit. However, transmission control has too many active users to be able to accept another. This condition may pass as other users sign off. If the condition persists, notify your SHARP APL representative so that steps may be taken to increase the system's resources.

P. 286, col. 2, para. 3: In the explanation of *VALUE ERROR* the second sentence should be expanded to read:

> This might be because you've used a dyadic defined function without a left argument. It might also be because the name is shadowed during execution of a function, or because the value really doesn't exist in the workspace.

P. 287, col. 1, paras. 2, 3: The references to ⎕*RUN* should be deleted from both of these paragraphs. Events 45 and 46 are trappable in the return code of ⎕*RUN*. The third sentence in paragraph 2 should read:

> If your reference to the workspace occurred in an APL expression (i.e. one that invoked ⎕*LOAD* or ⎕*QLOAD*), this event is trappable as event 45.

The second sentence in paragraph 3 should read:

> If your reference to the workspace occurred in an APL expression (i.e. one that invoked ⎕*LOAD* or ⎕*QLOAD*), this event is trappable as event 46.

P. 289, col. 1, para. 3: Amend the description of integer representation by substituting the following for the present last sentence:

> Each integer element is represented by 32 binary bits. A negative value is represented in 2-complement; when the first bit is 1, the represented number is negative.

P. 289, col. 1, last paragraph: The exact range of integers that can be precisely represented is slightly misstated. It is in fact numbers whose magnitude is less than 16⋆14, which is 72,057,594,037,927,936.

P. 290, col. 2, para. 2: The reference to ⋆ should say ↓ . The first sentence of that paragraph should then read:

> **Tolerance during automatic conversion to integer:** A number of functions require one or both of their arguments to be integral (for example, the left arguments of ↑ ↓ ⌽ , both arguments of ? , and the argument of monadic ⍳).

P. 295: In the list of scalar dyadic functions, the entry for *X*|*Y* should now show note 3 (showing that the function is subject to comparison tolerance, ⎕*CT*).

P. 301, Table of event numbers: Add the following events:

```
25 FILE SYSTEM HARDWARE ERROR
29 FILE LIBRARY NOT AVAILABLE
```

P. 302, description of dyadic ⎕*PACK*: The left argument may contain only a single name. The description should therefore refer to the left argument as *X* rather than *NL*, and should read as follows:

P←*X* ⎕*PACK Y*	Creates a package containing the array or package *Y* with the name specified in *X*.

P. 307, addition: Insert the following under 1 *WSFNS* in **Selected Public Workspaces**:

1 *DIRECTDEF*	Functions for direct definition

P. 309: The list of SHARP APL Technical Notes has been revised. The currently available titles are:

SATN-0	1 JAN 76		SATN Introduction
SATN-2	21 OCT 79	REV. 4	Control Messages
SATN-4	1 APR 78	REV. 2	N-tasks and B-tasks
SATN-5	1 FEB 80	REV. 3	Batch APL
SATN-8	1 MAR 79	REV. 2	*HSPRINT*
SATN-9	1 NOV 80	REV. 2	Usage Inquiry System
SATN-10	1 JUN 78	REV. 2	*SORTREQ*
SATN-19	1 JAN 77		*FILEPRINT*
SATN-22	15 JUL 81	REV. 4	APL Workspace Transfer
SATN-23	15 JUL 78	REV. 1	Comparison Tolerance
SATN-28	11 JUL 77		Terminal Control
SATN-29	15 JUN 78		System Time and Timestamps
SATN-34	10 SEP 80		Replication
SATN-35	20 SEP 80		Extended Upgrade and Downgrade
SATN-36	OCT 80		Direct Definition
SATN-37	1 JUN 81		IBM 3270 User Guide
SATN-38	15 JUL 81		PJAM User Guide
SATN-39	1 JUN 81		The SHARP APL S-task Interface
SATN-40	20 JUN 81		Complex Numbers
SATN-41	20 JUN 81		Composition and Enclosure

P. 311, col. 1: For the listing "active workspace: sharing between two," change page 91 to 83.

P. 313, col. 1: For the listing "auxiliary processor," change page 91 to 83.

P. 313, col. 2: The entry for "backspace" should read "character that (not than) can be displayed but not entered."

P. 314, col. 2: For the listing "B-task: shared-variable partner," change page 91 to 83.

P. 322, col. 1: The entry for "except" should read "action *N* (not *S*) in trap definition."

P. 323, col. 2: Insert the following listing between the listings for *FILE INTERRUPT* and **File library**:

FILE LIBRARY NOT AVAILABLE, 110, 281, 301

Insert the following listing between those for *FILE SYSTEM ERROR* and **File system independent of APL interpreter**:

FILE SYSTEM HARDWARE ERROR, 110, 281, 301

P. 323, col. 2: Insert "**File quota and file reservation limit**, 101" before "file quota, 101 216." Delete "**file reservations: Limit on total**, 101."

P. 328, col. 2: Add page 101 to the listing for "limit: file reservation" so that the third line from the bottom of the page reads:

file reservation, 101 216

Delete the last line, "**Limit on total file reservations**, 101."

P. 332, col. 1: For the listing "network: pairwise sharing," change page 91 to 83.

P. 333, col. 1: Remove *ERROR* from the listing for *OPEN QUOTE* so that it reads:

OPEN QUOTE: not a trappable event, 117

P. 333, col. 2: Delete the extra "s" from "accesss" in the listing, "ownership of a file: implicit last row of accesss matrix."

P. 335, col. 1: For the listing "processor: auxiliary," change page 91 to 83.

P. 335, col. 2, addition: Insert the following entry between "proposition: used to control compress and expand, 48" and "**Propositions on equality**, 135":

Proposition on set membership, 159

P. 339, col. 1: For the listing "⎕*WS*, using 2 ⎕*WS* or 4 ⎕*WS* with shared variable...etc.," change 2 ⎕*WS* to 6 ⎕*WS*.

Change "quota: file reservation total, 101" to **quota: file and file reservation limit**, 101.

Insert "quota:" before "number" on the following line so that the entry reads:

quota: number of files a task may have tied, 212.

P. 340, col. 1, addition: Insert an entry for *RESEND* as follows:

RESEND, 285

P. 342, col. 1: The entry under "shape of result:" should be "take and drop" (not "trop").

P. 342, col. 2: For the listing **Shared variables for communication between workspaces**, change page 91 to 83.

P. 345, col. 2: For the listing "terminal-task," change page 91 to 83, so that the page numbers read, "7 8 79 83 95 198 250 264 270."

P. 346, col. 2: For the listing "T-task, sharing with other user's," change page 91 to 83.

P. 349, col. 1: The last entry for "zero" should read, "value of ⎕*SC* (not ⎕*SVC*) when no clone ID established, 256."

Printed in the United States of America
Publication Code: 81RM06E