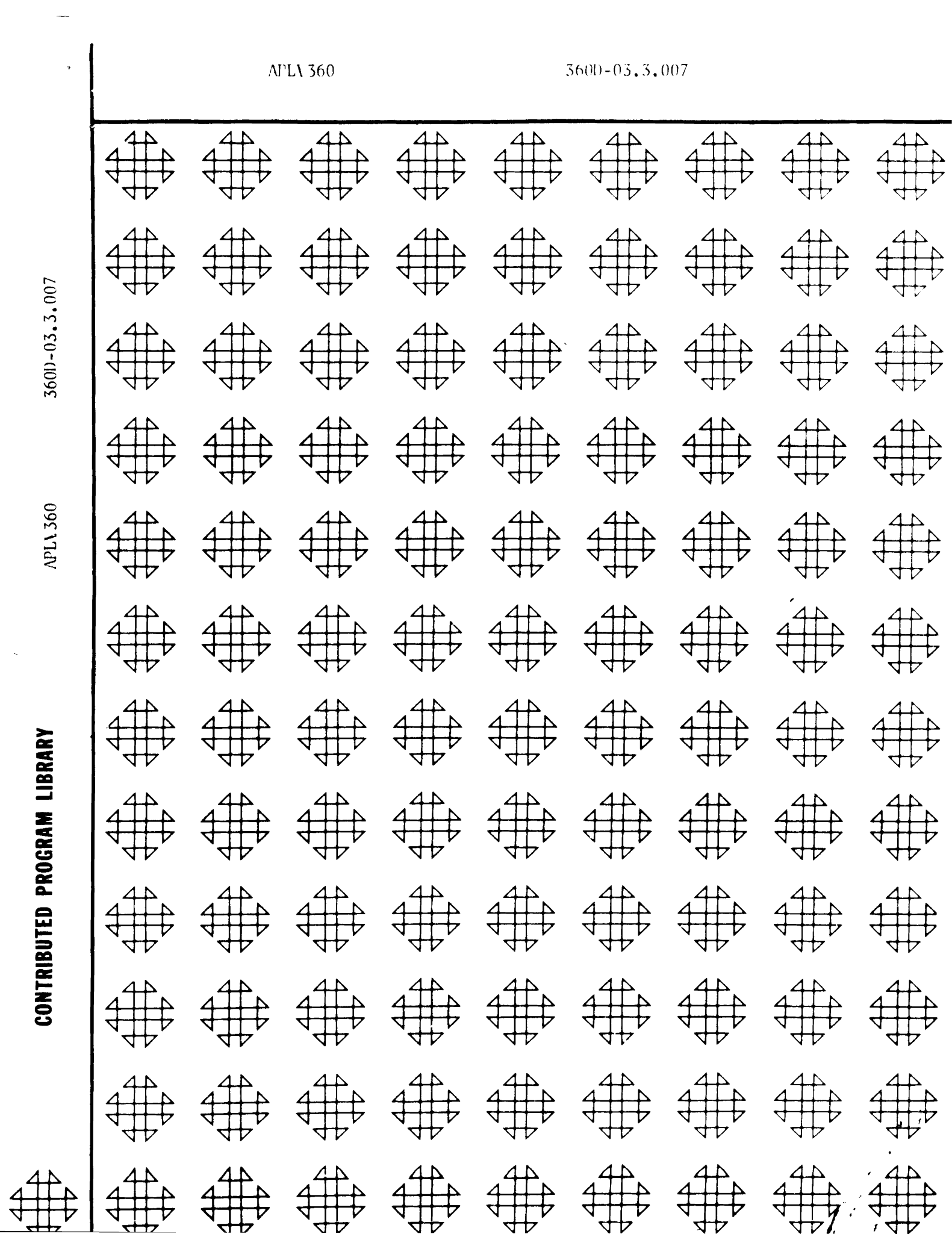
360D-03.3.007 APL\360

CONTRIBUTED PROGRAM LIBRARY



## DISCLAIMER

This program and its documentation have been contributed to the Program Information Department by an IBM employee and are provided by the IBM Corporation as part of its service to customers. The program and its documentation are essentially in the author's original form and have not been subjected to any formal testing. IBM makes no warranty expressed or implied as to the documentation, function, or performance of this program and the user of the program is expected to make the final evaluation as to the usefulness of the program in his own environment. There is no committed maintenance for the program.

Questions concerning the use of the program should be directed to the author or other designated party. Any changes to the program will be announced in the appropriate Catalog of Programs; however, the changes will not be distributed automatically to users. When such an announcement occurs, users should order only the material (documentation, machine readable or both) as indicated in the appropriate Catalog of Programs.

# APL\360

Author: L. M. Breed

Co-author: R. H. Lathwell

Address inquiries to:

L. M. Breed
IBM
P. O. Box 218
Yorktown Heights, New York, 10598

© International Business Machines Corporation, 1968

TABLE OF CONTENTS

Program Abstract 3

Preface 4

Magnetic Tape Key 9

Volume I: APL\360: User's Manual 12

Authors: A. D. Falkoff and K. E. Iverson

Volume II: APL\360: Operator's Manual 119

Author: R. H. Lathwell

Volume III: APL\360: System Generation and Maintenance 172

Author: R. H. Lathwell

Detailed tables of contents are at the beginning of each volume.

Program Abstract

APL\360 is a conversational time-sharing system based on a mathematical programming language first defined by K. E. Iverson. The language is concise and has a simple syntax. It has a large set of primitive operations which work directly on arrays. The implementation provides a simple immediate-execution mode and a convenient program definition facility. It has fast response, and uses succinct diagnostic messages. It provides the ability to save work between sessions, to create programming packages, and to exchange programs and data between users. Uses of the system include mathematical and statistical calculation, symbol manipulation, and general data processing. It has been used extensively in computer-related instruction, and in the design of hardware and software.

Minimum configuration: 256K storage; universal instruction set; a 2314 or three 2311's; one 2702 or 2703; and 2741 or 1050 terminals. Additional storage, disks, and transmission controls are supported. A typical Model 50 configuration supports 60 terminals. The system is written in Assembly language, incorporates a modified DOSIII, and allows standard DOS background operation.

PREFACE

APL\360 is a conversational terminal system that has been operating within IBM since the Fall of 1966. Conceived as an experimental system for exploring certain aspects of computer science, its purpose required that it operate under a realistic load in an environment that was not artificially constrained. To this end, the members of the IBM Research staff, and others, were encouraged to learn the APL language and use the APL system. It was not anticipated that it would have the impact that it did:

In twenty months of regularly scheduled operation, clocking well over 100,000 terminal-hours of use, the availability of APL\360 has materially changed the computing habits of the Research organization.

Heavy users of batch processing have turned to APL for on-line development of their algorithms.

Many laboratory data reduction chores formerly done by batch operation or desk calculator are now executed in a timely way at local terminals, using locally written, stored APL programs.

Automatic collection of experimental data has, in many instances, been put on-line to APL.

Routine correspondence and technical papers are prepared at terminals, with the help of text-handling programs written in APL.

Many professional staff members who formerly were indifferent to, or actively resisted, the use of computers, have become steady users of the APL system.

In addition to IBM Research personnel, use of the system was offered to other locations within the IBM Company, and it was also used experimentally in elementary and secondary schools and in universities. The general findings may be summarized as follows:

Although APL is easy for a beginner to learn, non-programmers (as well as programmers) often develop an interest in sophisticated use of the language, because of its analytical power and mathematical structure.

The primitive array operations of APL make it a good language for scientific problems, text handling, and general data processing, because arrays are fundamental to all of these applications.

The use of a powerful, readily accessible computational facility can materially change the quality and orientation of an academic course.

Other things being equal, acceptance of time-sharing as a general mode of operation is strongly dependent upon its reliability and availability -- regularly scheduled hours are essential, and the more the better, including nights and weekends.

System features. APL 360 is based upon APL, the language first defined by K. E. Iverson in A Programming Language (John Wiley, 1962). It is an interpretive time-sharing system that builds upon the array operations and structural integrity of APL to provide a running system with the following salient characteristics:

Simple, uniform rules of syntax

Use of common symbols for the ordinary arithmetic operations

Free-form decimal input

A large set of primitive operators

Use of defined functions (programs) with the same facility and syntactic variety as primitive operators

Automatic internal conversions of data representation, with full double-precision arithmetic (16 decimal digits) when required

Very fast response

A library structure built around workspaces that hold both programs and data

An immediate-execution mode completely free of irrelevant keywords

A comprehensive, integrated set of system commands for managing workspaces and libraries, and for other essential functions

Three levels of security; account numbers, workspaces, and programs can be individually locked against use or display

Visual fidelity between hard copy and transmitted entries, which ensures reproducibility of results

Succinct diagnostic reports

Hardware and performance. The APL\360 system, comprising a time-sharing supervisor and an APL interpreter, runs in concert with a modified DOS Version 3, which appears the same as the standard DOS release of the corresponding change level. During APL operation it requires an S/360 machine with at least an H-level memory, a transmission control unit, and three disk drives.

The particular configuration used in the original Research experiment is a Model 50H with a 2703 Transmission Control Unit and a 2314 Direct Access Storage Facility. This system has 61 ports, and with about that number in use seems to be a well-balanced system with regard to relative usage of the central processor and input-output equipment.

APL 360 has run (or is now running) experimentally on S/360 Model 40 and Model 65 machines, and will run on all larger models. It will also run on an S/360 Model 44 equipped with the Trap and Emulate RPQ. It supports IBM 2741 and 2740-1 Communications Terminals, and 1050 Teleprocessing Terminals with or without paper tape or punched-card equipment.

Average reaction time of the Model 50 system (i.e., time to respond to trivial requests from a terminal) peaks at about three-tenths of a second. With up to 40 ports in use, most such responses are essentially instantaneous; when fully loaded, there are occasional delays of as much as four seconds.

The time for serving non-trivial requests naturally varies according to the extent to which the CPU must be shared during the computation. Because the primitive operations of APL are defined on arrays, relatively little interpretive overhead is needed for many large computations, and the actual CPU time used for a typical computation may run from 20 to 50 times that for efficient compiled code; but the overall efficiency is likely to be comparable, if not superior, to batch processing in many applications if the usual compiling and loading times for batch work are taken into account. If debugging time is included, the advantage of interpretive APL becomes even greater.

Applications. The workspace-centered organization of APL\360 makes it very convenient to assemble application packages, because a collection of programs and data in a workspace moves as a coherent unit whenever the workspace containing it is stored or activated. Some of the packages that have been developed on the experimental systems so far are:

A comprehensive collection of functions for statistical calculations

A coordinate geometry package that includes a position plotter

A management information system for controlling the allocation of manufacturing resources

A text-editing and composing facility that has been used for preparing this document, among others

An algebraic manipulator that simplifies polynomials and operates on matrices of algebraic expressions

A bookkeeping system for balancing accounts

Many collections of programs for numerical analysis of both real and complex functions

The IBM Research Model 50 APL\360 system serves more than 700 active users whose work ranges over many fields: from simple desk calculations to experimental investigations of semi-groups, from laboratory data reduction to theoretical physics, from the working out of small programming problems to the design and modelling of major systems, and from administering arithmetic drill to elementary-school children to use in graduate-level engineering courses.

Documentation. The accompanying document consists of three volumes bound together: User's Manual, Operator's Manual, and System Generation and Maintenance.

MAGNETIC TAPE KEY

FOR 2311 BASED SYSTEM:

One (1) volume containing the listed files and three (3) tape marks, arranged as follows:

800 bpi; 9 track; EBCDIC

| | |
|---|---|
| IPL record | 24 bytes |
| Disk initialization program | variable length records (range 24 to 330 bytes) unblocked |
| T/M | |
| IPL record | 24 bytes |
| Disk restore program | 80 byte records unblocked |
| APL 360 | variable length records (range 15 to 2000 bytes) unblocked |
| T/M | |
| T/M | |

This volume contains 21009 records



FOR 2314 BASED SYSTEM:

Two (2) volumes composed as follows:

Volume 1 -- contains the listed files and two (2) tape marks arranged as follows:

800 bpi; 9 track; EBCDIC

| | |
|---|---|
| IPL record | 24 bytes |
| Disk initialization program | variable length records (range 24 to 330 bytes) unblocked |
| T/M | |
| IPL record | 24 bytes |
| Disk restore program | 80 byte records unblocked |
| APL\360 | variable length records (range 24 to 2000 bytes) unblocked |
| T/M | |

This volume contains 23549 records

Volume 2 -- contains the listed file and two (2) tape marks arranged as follows:

800 bpi; 9 track; EBCDIC

| | |
|---|---|
| APL 360 system (continued) | 5606 variable length records (range 15 to 175 bytes) unblocked |
| T/M | |
| T/M | |

FOR BOTH SYSTEMS

One (1) volume containing the specified file and four (4) tape marks arranged as follows:

800 bpi; 9 track; EBCDIC

Standard header label

T/M

APL 360 starter library — 56 varible length records (range 24 to 3500 bytes) unblocked

T/M

Trailer label

T/M

T/M



Volume I

APL\360: User's Manual

A. D. Falkoff

K. E. Iverson

© International Business Machines Corporation, 1968

## ACKNOWLEDGEMENTS

The APL language was first defined by K. E. Iverson in A Programming Language (Wiley,1962) and has since been developed in collaboration with A. D. Falkoff. The APL*360 Terminal System was designed with the additional collaboration of L. M. Breed, who, with R. D. Moore*, also designed the S/360 implementation. The system was programmed for S/360 by Breed, Moore, and R. H. Lathwell, with continuing assistance from L. J. Woodrum*, and contributions by C. H. Brenner, H. A. Driscoll‡, and S. E. Krueger‡. The present implementation also benefitted from experience with an earlier version, designed and programmed for the IBM 7090 by Breed and P. S. Abrams**.

The development of the system has also profited from ideas contributed by many other users and colleagues, notably E. E. McDonnell, who suggested the notation for the signum and the circular functions.

In the preparation of the present manual, the authors are indebted to L. M. Breed for many discussions and suggestions; to R. H. Lathwell, E. E. McDonnell, and J. G. Arnold ¬ for critical reading of successive drafts; and to Mrs. G. K. Sedlmayer and Miss Valerie Gilbert for superior clerical assistance.

A special acknowledgement is due to John L. Lawrence, who provided important support and encouragement during the early development of APL implementation, and who pioneered the application of APL in computer-related instruction.

* I. P. Sharp Associates, Toronto, Canada.
Systems Architecture, IBM Corporation, Poughkeepsie, N.Y.
‡ Science Research Asosociates, Chicago, Illinois.
** Computer Science Department, Stanford University, Stanford, California.
¬ Industry Development, IBM Corporation, White Plains, N.Y.

## TABLE OF CONTENTS

PART 1: GAINING ACCESS 13

PHYSICAL EQUIPMENT 19
Preferred features
THE APL CHARACTER SET 20
Use of other character sets
THE RECORDING TERMINAL 21
ESTABLISHING A CONNECTION 21
Set up terminal, Dial computer
ENTRIES FROM THE KEYBOARD 23
Transmission signals, Mistakes, Transmission errors, Special features of IBM 1050 terminals
STARTING AND ENDING A WORK SESSION
Sign-on, Limited use of the system, Disconnect dial-up connection, Break any connection

PART 2: SYSTEM COMMANDS 29

WORKSPACES AND LIBRARIES 29
Workspaces, Libraries
NAMES
Local and global significance
LOCKS AND KEYS
ATTENTION 32
USE OF SYSTEM COMMANDS 33
Classification of commands, Normal response and trouble reports, Clear workspace, Summary
TERMINAL CONTROL COMMANDS
Forced endings, The CONTINUE workspace, Interrupted activities, Detailed description
WORKSPACE CONTROL COMMANDS
Application Packages, Groups, Detailed description
LIBRARY CONTROL COMMANDS
Continuity of work, Workspace identification, Library and account numbers, Storage allotment, Use of the CONTINUE workspace, Purging a workspace, Detailed description
INQUIRY COMMANDS
User codes, Detailed description
COMMUNICATION COMMANDS
Detailed description

PART 3: THE LANGUAGE


FUNDAMENTALS 64
Statements, Scalar and vector constants, Names and spaces, Overstriking and erasure, End of statement, Order of execution, Error reports, Names of primitive functions

SCALAR FUNCTIONS 68
Monadic and dyadic functions, Vectors, Index generator

DEFINED FUNCTIONS 72
Introduction, Branching, Local and global variables, Explicit argument, Explicit result, The forms of defined functions, Use of defined functions, Recursive function definition, Trace control

MECHANICS OF FUNCTION DEFINITION 79
Labels, Revision, Display, Line editing, Reopening function definition, Locked functions, Deletion of functions and variables, System command entered during function definition

SUSPENDED FUNCTION EXECUTION 82
Suspension, State indicator, Stop control

HOMONYMS 84
Variable names, Function names

INPUT AND OUTPUT 85
Evaluated input, Character input, Escape from input loop, Normal output, Heterogeneous output

RECTANGULAR ARRAYS 88
Introduction, Vectors dimension catenation, Matrices dimension ravel, Reshape, Uses of empty arrays, Indexing, Indexing on the left, Index origin, Array output

FUNCTIONS ON ARRAYS 95
Scalar functions, Reduction, Inner product, Outer product

MIXED FUNCTIONS 99
Introduction, Transpose, Monadic transpose, Rotate, Reverse, Compress, Expand, Decode, Encode, Index of, Membership, Take and drop, Grade up and down, Deal, Comments

MULTIPLE SPECIFICATION 107

SYSTEM DEPENDENT FUNCTIONS 108

PART 4: LIBRARY FUNCTIONS 111

APPENDIX A 120
SAMPLE TERMINAL SESSION

APPENDIX B 135
ADVANCED EXAMPLES

BIBLIOGRAPHY 148


LIST OF ILLUSTRATIONS


| | | Page |
|---|---|---|
| Table 1.1 | RECOMMENDED FEATURES AND OPTIONS FOR TERMINALS | 19 |
| Figure 1.2 | APL\360 KEYBOARD | 20 |
| Table 1.3 | TELEPHONE NUMBERS | 22 |
| Table 2.1 | SYSTEM COMMANDS | 35 |
| Table 3.1 | ERROR REPORTS | 67 |
| Table 3.2 | PRIMITIVE SCALAR FUNCTIONS | 69 |
| Table 3.3 | FORMS OF DEFINED FUNCTIONS | 77 |
| Table 3.4 | DIMENSION AND RANK VECTORS | 89 |
| Table 3.5 | IDENTITY ELEMENTS OF PRIMITIVE SCALAR DYADIC FUNCTIONS | 96 |
| Table 3.6 | INNER PRODUCTS FOR PRIMITIVE SCALAR DYADIC FUNCTIONS f AND g | 98 |
| Table 3.7 | OUTER PRODUCTS FOR PRIMITIVE SCALAR DYADIC FUNCTION g | 98 |
| Table 3.8 | PRIMITIVE MIXED FUNCTIONS | 100 |
| Table 3.9 | TRANSPOSITION | 102 |
| Table 3.10 | SYSTEM DEPENDENT FUNCTIONS | 109 |

This page has been intentionally left blank.

# PART 1

# GAINING ACCESS

An APL\360 System comprises a central computer and an indefinite number of typewriter-like terminals. A certain number of these terminals may be simultaneously linked to the computer, according to the number and type of access ports on the computer.

This part of the manual describes the terminal equipment required for interacting with the system, tells how to establish a connection between a terminal and the central computer, and gives, in simplest form, the procedures for starting and ending a work session.

## PHYSICAL EQUIPMENT

A remote terminal for use with the system must be either an IBM 2741 Communications Terminal, an IBM 2740-1 Communications Terminal equipped with the Transmit Control feature, or an IBM 1050 Teleprocessing Terminal. It may connect to the central computer through the dial-up telephone network, by a leased telephone line, or by private wire.

Dial-up connections are effected by means of a Western Electric Dataset #103A-2 or the equivalent, or a compatible acoustic coupler. A leased telephone line connection requires the use of a Western Electric Dataset #103F-2 or the equivalent. A direct-wired connection is effected by means of an appropriate IBM line adapter (modem). In the last case, two-wire connections should be avoided, if possible, since their use rules out an interrupt facility.

Preferred features. The APL\360 system will work with many variations of the terminal types given above, but certain features and options are desirable. Dial-up connections provide the greatest flexibility, both in overall system configuration, and in certain details of operation. Similarly, although the APL printing element is based on a 12-pitch font, and is available in both Selectric® and PTTC/BCD keyboard encoding (i.e., the correspondence between

| FEATURE OR OPTION | 1050 | 2740-1 | 2741 |
|---|---|---|---|
| Control Unit | 1051-2 | | |
| Voltage (115 AC), Non-lock plug | 9881 | 9881 | 9881 |
| Dataset Attachment | 9114 | 9114 | 9114 |
| Dial Up | NR | 3255 | 3255 |
| Transmit Control | NR | 8028 | NR |
| Automatic EOB | RPQ E27283 | Do not use | NR |
| Typamatic Keys | NA | NA | 8341 |
| Interrupt | RPQ E27428 | RPQ F17913 | 4708 |
| Text Time-out Suppression | 9698 | NR | NR |
| First Printer Attachment | 4408 | NR | NR |
| Automatic Ribbon Shift Select | 1295 | NA | NA |
| Typing Table | 9705 | NR | NR |
| Printer-Keyboard | 1052-2 | | |
| APL Printing Element, PTTC/BCD | 1167988 | 1167988 | 1167988 |
| or Standard Selectric® | NA | 1167987 | 1167987 |
| Keys, APL Keyboard | RPQ M40174 | RPQ M40174 | RPQ M40174 |
| Character Spacing, 10 per inch | 9104 | 9104 | 9104 |
| Line Feeding, 6 per inch | 9435 | 9435 | 9435 |
| Accelerated Carrier Return | 1006 | NA | NA |

Notes. NR: feature is standard equipment, or is not required.
NA: not available (July 1968).
The numbers are IBM-domestic identifications.

Table 1.1: RECOMMENDED FEATURES AND OPTIONS FOR TERMINALS

keyboard layout and character positions on the printing element), specification of 10-pitch character spacing and Selectric® encoding will allow a greater variety of printing elements to be used with the terminal. While it is not essential, the convenience of having the interrupt feature cannot be overestimated.

Paper tape equipment (1054-1 Reader and 1055-1 Punch) and punched-card equipment (1056-1 Reader and 1057-1 Punch) can be used with IBM 1050 terminals. The punched-card facilities should have Extended Character features 3861 and 3860, for reader and punch, respectively.

IBM identifications for recommended terminal features and options are given in Table 1.1. Complete specifications for terminals, and information on other options, should be obtained from local IBM representatives.

Figure 1.2: APL\360 KEYBOARD

THE APL CHARACTER SET

The APL\360 keyboard is shown in Figure 1.2. The numerals, alphabetic characters, and punctuation marks appear in their usual places, although the alphabet is used in only a single case: letters print as upper-case italics, but are produced only when the keyboard is in lower-case position (i.e., not shifted).

The special characters, most of which are produced with the keyboard shifted, generally have some mnemonic connection with their alphabetic or numeric correspondents. This may be appearance (ω over *W*), Greek-Roman equivalence (ρ over *R*), sequence (< ≤ = ≥ > ≠ over 3 4 5 6 7 8), or some -- possibly far fetched -- relationship between the APL function represented by the symbol and the letter (* over *P* for power, ' over *K* for "kwote", and ⌈ over *S* for ceiling).

Use of other character sets. The part numbers of APL printing elements are given in Table 1.1. However, any printing element may be used with the APL system, since the encoded characters generated by the keyboard and transmitted to the computer are independent of the particular element mounted on the terminal. Subject to programmed intervention, the transmitted information will always be interpreted according to the APL keyboard characters.

Non-APL printing elements are frequently useful in conjunction with special-purpose APL programs designed to exploit their character sets. Also, any element that matches the keyboard encoding (Selectric® or PTTC/BCD) of the terminal can be used for straightforward numerical work, since letters and digits print properly with such elements. The visual interpretation of complex APL expressions is, of course, awkward with any but an APL printing element.

## THE RECORDING TERMINAL

As connections with remote terminals are established and broken, and users start and end work sessions, a printed record of this traffic is generated at the system's recording terminal. This terminal, which is usually, but not necessarily, located at the central computer site, is ordinarily attended by an APL Operator who monitors the operation of the system, and provides a common point of contact for users.

There are certain supervisory functions, essential to the operation of APL\360, which can be effected only from the recording terminal. Thus, this terminal holds a privileged position relative to others. The enrollment of new users, and the allocation of library space, are examples of this kind of function.

## ESTABLISHING A CONNECTION

The directions that follow assume the use of a dial-up connection with a dataset. Instructions for the use of acoustic couplers should be obtained from their suppliers. Where terminals are connected to the computer by leased lines or private wires, instructions on dialing procedure (EC2) are irrelevant, but local sources of information should be consulted for equivalent procedures.

| ACTION | | NOTES |
|---|---|---|
| EC1. Set up terminal: Insert paper, mount an APL printing element, connect terminal to power source, and set switches as follows: | | |
| IBM 2741 or 2740 Terminal | | |
| LCL/COM | COM | The power switch is at the right of the keyboard. On 2741's, the LCL/COM switch is on the left side of the terminal stand, toward the rear; on 2740's, it is to the right of the power switch. |
| Power | ON | |

IBM 1050 Terminal

| | | |
|---|---|---|
| ATTEND/UNATTEND | ATTEND | Not all 1050's have all switches; those present must be set as indicated. The states of switches not listed here are immaterial. |
| Keyboard | SEND | |
| Printer | SEND-RECEIVE | |
| Reader 1 | OFF | |
| Punch 1 | OFF | |
| EOB | MANUAL | If it is known that RPO E27283 (see Table 1.1) is installed, set the EOB switch to AUTO. |
| Line test | OFF | |
| Single step | OFF | |
| Line control | ON | The line control switch is inside the rear door of the 1051 Control Unit. The power switch is on the left side of the control unit, toward the front. |
| Power | ON | |

On 2741 and 2740 terminals, test to see if the keyboard is locked by trying the shift key. If the key is operable, press the carrier return and test again.

If the keyboard does not lock after a carrier return, check the switches and try once more. If the switches are set properly and the keyboard remains unlocked, the terminal is faulty.

EC2. Dial computer:
Set the telephone pushbutton switch to TALK and follow ordinary dialing procedure. After two rings, at most, the telephone will respond with a steady, high-pitched tone.

Telephone numbers are given in Table 1.3. If the line is busy, try a different number or call the APL Operator to inquire about an open line.

| | |
|---|---|
| 123 456-7890 | 123 456-7890 |
| Insert a table of access telephone numbers here. An assistance number should be included. | |
| APL Operator: | 123 456-7890 |

Table 1.3: TELEPHONE NUMBERS

| | |
|---|---|
| Promptly set the pushbutton switch to DATA by holding the DATA button down firmly for a moment and then releasing. | The DATA button should light, and will remain lit as long as the terminal is connected to the computer. If it does not light, check the power connection to the dataset. If it lights, but quickly goes out, check the power connection to the terminal, the cable connection to the dataset, and the switch settings on the terminal. Then retry from EC1. |
| Cradle the handset. | |

Response: The keyboard will unlock, indicating that the computer is ready to accept an entry from the terminal.

The connection established by the foregoing procedure is only tentative, and will be broken by the central computer if further communication does not take place within 60 seconds. Therefore, the next step -- the sign-on procedure (EC3) given below -- should be executed promptly.

ENTRIES FROM THE KEYBOARD

After a connection is established, normal communication between a terminal and the central computer is carried on by means of entries from the typewriter keyboard, which alternately locks and unlocks as each entry is made and the computer completes its work. The general procedure is to type an instruction or command, strike the carrier return to indicate the end of the message, and follow this by a transmission signal.

Transmission signals. The transmission signal is generated differently, according to the terminal type and its equipment:

2741. A transmission signal is automatically generated in the proper sequence (i.e., after the carrier return signal) when the RETURN key is struck.

2740. The transmission signal is produced by striking the EOT key after the RETURN key. (Do not use the the EOB key, or the automatic EOB feature available on these terminals.)

1050. On terminals equipped with an automatic EOB RPQ (see Table 1.1), the transmission signal is produced automatically when the RETURN key is struck. Otherwise, an EOB must be produced manually, by striking the numeral-5 key, while the key marked ALTN CODING is held down. (Note that the automatic EOB feature available for 1050 terminals cannot be used with APL\360.)

A transmission signal does not cause a character to be printed, and its omission will therefore be evidenced only by the state of the terminal: the keyboard will remain unlocked, and no response will be forthcoming from the system.

In the remainder of this manual the need for carrier return and transmission signal will not be explicitly mentioned, since they are required for every entry.

Mistakes. Before the carrier return (and transmission signal) that completes an entry, errors in typing can be corrected as follows: backspace to the point of error and then depress the linefeed button (marked ATTN on 2741 terminals). This will have the effect of erasing everything to the right of, and including, the position of the carrier. The corrected text can be continued from that point, on the new line.

This procedure can be used at any time once the sign-on (EC3) has been accomplished. In case of error in the sign-on itself, the entry should be made as is. The system will provide an appropriate trouble report, following which, a correct entry may be made.

Transmission errors. There are occasional transient failures in the communication between a terminal and the central computer. If the failure occurs during the transmission from the terminal, the system will respond with a resend signal: on 1050 terminals, the RESEND warning light will go on, and on other terminals the message *RESEND will be printed. In any case, the last entry from the keyboard must be repeated. The warning light on the 1050 should first be extinguished by pressing the adjacent button.

Failures in the other direction are usually evidenced by the appearance of a spurious character, whose presence in the printed output is obvious in most contexts. However, there is no absolutely certain way of detecting such a failure.

**Special features of IBM 1050 terminals.** The keyboard of a terminal equipped with a REQUEST button will not unlock, when it otherwise should, until the button is depressed. On terminals equipped with a timer, the keyboard will lock before an entry is completed if approximately 18 seconds have elapsed since the last keyboard action. Locking can be forestalled by occasionally striking the shift key, but if it does happen, the keyboard can be forced to unlock by flipping the line-control switch inside the 1051 Control Unit to OFF, and back to ON.

If a terminal is to be used exclusively with APL\360, the Keyboard Request feature should be removed, and the Text Time-out Suppression feature should be added.

## STARTING AND ENDING A WORK SESSION

Each user of the system is assigned an account number. This number is used to effect the sign-on that initiates a work session; serves to partially identify any work that the user may store in the system between sessions; and is used for accounting or billing purposes.

If the account number is not known, or if one of the trouble reports given below is encountered and not understood, a message of inquiry can be sent to the APL Operator. This is accomplished by entering )OPR followed by a space and one line (not exceeding 120 characters) of an appropriate text.

Such a message can be sent at any time after a connection has been established. It causes the keyboard to lock, awaiting a reply. If no reply is forthcoming, (and the sign-on has not been completed), the connection will have to be broken and re-established before further communication with the system is possible. (After the sign-on, the keyboard may be unlocked by an attention signal, described in Part 2.)

| ACTION | NOTES |
|---|---|
| EC3, Sign on:<br>Enter )<br>followed by an account number, with a key (i.e., a colon and password), if required. | The use of passwords as locks and keys is described in Part 2. A new user will have been advised if a key is required for his first sign-on. |

Effect:
1. A workspace will be activated for the terminal.

2. Accumulation of time charges will begin.

Response:
1. A broadcast message from the APL Operator may be printed.

2. The port number, time of day, date, and user name associated with the account number will be printed on one line. The system identification will be printed on another line.

A workspace can be thought of as both a notebook and a scratch pad. The details are explained in Part 2.

Trouble reports:

*INVALID ACCOUNT NUMBER*
means either exactly what it says, or that the number has a lock associated with it and the wrong key was used. The APL Operator should be consulted if help is required.

*INCORRECT SIGN-ON*
means the form of the transmitted command was faulty. Retry with a properly formulated sign-on.

*ALREADY SIGNED ON*
means that a work session is already in progress at the terminal. To start a session with a different account number, use command TC5 (see Part 2), which ends the current session but holds the connection, and retry from the beginning of EC3.

*[illegible]*
means just that, or a temporary condition due to delays in the central computer. Retry from EC2 after two minutes. If the condition persists, notify the APL Operator.

*[illegible]*
means that authorization for use of that number has been withdrawn.

| | |
|---|---|
| 3. *SAVED*, followed by the time of day and date that the activated workspace was last stored. | This response will be omitted if the activated workspace is clear (i.e., not holding information). If the response is given, the workspace is named *CONTINUE*. The use of workspace names is explained in Part 2. |
| 4. The keyboard will be unlocked. | If this is the only reponse, a transmission error has occurred, or the entry did not start with an APL right parenthesis. In either case, the entry should be repeated in correct form. If the condition persists, retry from EC2, possibly dialing a different number. |

A work session is started, and the full APL system becomes available, once the sign-on is accomplished. Any system command of Part 2 or APL operation of Part 3 may now be entered for execution.

Limited Use of the System. No system command other than the sign-on given here is required in order to make use of Part 3, and the reading of Part 2 may therefore be deferred if only casual or restricted use is to be made of the system. For the purposes of such use, a work session may conveniently be terminated by one of the following procedures:

| ACTION | NOTES |
|---|---|
| EC4. Disconnect dial-up connection:<br>Set the power switch to OFF. | Use this procedure for dial-up connections only. |
| Effect:<br>1. The active workspace will be stored under the name *CONTINUE*. | If the workspace is clear, it will not be stored at this time. If it is stored, it will be automatically re-activated when the same account number is next used to sign on. See note for EC3, Response 3. |
| 2. The duration of the work session and the amount of computer time used will be noted internally for later accounting. | |
| 3. The connection to the central computer will be broken. | |
| Response: None. | The DATA light will go out. |
| EC5. Break any connection:<br>Enter *)CONTINUE* | This is command TC4, detailed in Part 2. |
| Effect:<br>1. 2. and 3. The same as for EC4. | |
| Response:<br>1. Time of day and date, followed by *CONTINUE* | Trouble reports:<br>*NOT WITH OPEN DEFINITION*<br>*INCORRECT COMMAND*<br>The meanings of these reports, and corrective actions for them, are given in Part 2. |
| 2. The port number, time of day, date, and user code will be printed. | User codes comprise three characters which partially identify users. Their use is explained in Part 2. |
| 3. Accounting information will be printed. | If a dial-up connection is being used, the DATA light will go out. |

PART 2

SYSTEM COMMANDS

APL operations deal with transformations of abstract objects, such as numbers and symbols, whose practical significance, as is usual in mathematics, depends upon the (arbitrary) interpretation placed upon them. System commands in the APL\360 System, on the other hand, have as their subject the structures which comprise the system, and control functions and information relating to the state of the system, and therefore have an immediate practical significance independent of any interpretation by the user.

In this Part the structure of the APL\360 system is described, and various notions essential to the understanding of system commands are introduced. Finally, the complete set of system commands is described in detail.

## WORKSPACES AND LIBRARIES

Workspaces. The common organizational unit in the APL\360 system is the workspace. When in use, a workspace is said to be active, and it occupies a block of working storage in the central computer. The size of the block, which is preset at a fixed value for a given system, determines the combined working and storage capacity of each workspace in that system. Part of each workspace is set aside to serve the internal workings of the system, and the remainder is used, as required, for storing items of information and for containing transient information generated in the course of a computation.

An active workspace is always associated with a terminal during a work session, and all transactions with the system are mediated by it. In particular, the names of variables (data items) and defined functions (programs) used in calculations always refer to objects known by those names in the active workspace; information on the progress of program execution is maintained in the state indicator of the active workspace; and control information affecting the form of output is held within the active workspace.

Libraries. Inactive workspaces are stored in libraries, where they are identified by arbitrary names. They occupy space in secondary storage facilities of the central computer and cannot be worked with directly. When required, copies of stored workspaces can be made active, or selected information may be copied from them into an active workspace.

Libraries in APL\360 are either private or public. Private libraries are associated with individual users of the system, and are identified by the user's account number. Access to them by other users is restricted in that one user may not store workspaces in another person's library, nor can he obtain a listing of the workspaces already stored there. However, one user may activate a copy of another user's (unlocked) workspace if he knows the library number and workspace name.

Public libraries are identified by numbers below 1000. They are not associated with individual users, although certain ones may be reserved by general agreement for groups of people working cooperatively. Anyone may store workspaces in a public library, and a listing of workspace names is available upon request if the library number is known. However, a workspace stored in a public library remains under the control of the user who put it there, and cannot be altered by others.

## NAMES

Names of workspaces, functions, variables, and groups (see workspace control commands) may be formed of any sequence of alphabetic (A to Z, and Δ to _) and numeric (0 to 9) characters that starts with an alphabetic and contains no blank. Only the first 11 characters of workspace names, and the first 77 characters of other names are significant. Longer names may be used, but additional characters beyond these limits are ignored.

The environment in which APL operations take place is bounded by the active workspace. Hence, the same name may be used to designate different objects (i.e., groups, functions, or variables) in different workspaces, without interference. Also, since workspaces themselves are never the subject of APL operations, but only of system commands, it is possible for a workspace to have the same name as an object it holds. However, the objects within a workspace must have distinct names, except as explained below.

Local and global significance. In the execution of defined functions it is often necessary to work with intermediate results which have no significance either before or after the function is used. To avoid cluttering the workspace with a multitude of variables introduced for such transient purposes, and to allow greater freedom in the choice of names, the function definition process (see Part 3) provides a facility for designating certain variables as local to the function being defined. Variables not so designated, and all functions and groups, are said to be global.

A local variable may have the same name as a global object, and any number of variables local to different functions may have the same name.

During the execution of a defined function, a local variable will supersede a function or global variable of the same name, temporarily excluding it from use. If the execution of a function is interrupted (leaving it either suspended, or pendent, see Part 3), the local variables retain their dominant position, during the execution of subsequent APL operations, until such time as the halted function is completed. System commands, however, continue to reference the global homonyms of local variables under these circumstances.

## LOCKS AND KEYS

Stored workspaces and the information they hold can be protected against unauthorized use by associating a lock, comprising a colon and a password of the user's choice, with the name of the workspace, when the workspace is stored. In order to activate a locked workspace or copy any information it contains, a colon and the password must again be used, as a key, in conjunction with the workspace name. Listings of workspace names, including those in public libraries, never give the keys, and do not overtly indicate the existence of a lock.

Account numbers can be similarly protected by locks and keys, thus maintaining the security of a user's private library and avoiding unauthorized charges against his account.

Passwords for locks and keys may be formed of any sequence of alphabetic and numeric characters up to eight characters long, without blanks. Characters beyond the eighth are ignored. In use as either a lock or key, a password follows the number or name it is protecting, from which it is set off by a colon.

## ATTENTION

Printed output at a terminal can be cut off, or the execution of an APL operation can be interrupted, and control returned to the user, by means of an attention signal. Since the keyboard is locked during printing or computing, the signal must be generated by means other than one of the standard keys.

On terminals equipped with an interrupt feature, the attention signal is generated by depressing the appropriate key once, firmly. On IBM 2741 terminals this key is usually of a distinctive color, and is marked ATTN. (The same key is used for linefeed when the keyboard is not locked.)

For terminals not so equipped, the attention signal is generated by momentarily interrupting the connection to the central computer. The method depends upon the type of connection:

> with dial-up telephones, uncradle the handset, set the pushbutton switch to TALK for two to three seconds, and then reset it to DATA;
>
> with leased telephone lines, set the terminal power switch to OFF and then back to ON, with deliberate speed.

If the connection is broken, in either case, for more than five seconds, the central computer will interpret this as a signal to end the work session and will execute action EC4 of Part 1.

Following an attention signal the keyboard will unlock, and the type carrier will return to the normal position for input (six spaces from the left margin). If the carrier does not do this, enter blank lines repeatedly until it does. In some cases a line will be printed before the keyboard unlocks, telling where a function in progress was interrupted.

Except for communication commands (and then only if the delivery of a message is delayed), the execution of system commands, once entered, cannot be interrupted. However, the printed responses or trouble reports following a system command can be suppressed by a properly timed attention signal.

## USE OF SYSTEM COMMANDS

System commands and APL operations are distinguished functionally by the fact that system commands can be called for only by individual entries from the keyboard, and cannot be executed dynamically as part of a defined function. They are distinguished in form by the requirement that system commands be prefixed by a right parenthesis, which is a syntactically invalid construction in APL.

There is some system control which it may be desirable to exert dynamically, and there are some items of system information which can be profitably used during the execution of a program. For these purposes APL\360 provides appropriate system-dependent functions and library functions, which can be used like other APL operations. These functions are described in Part 3 and Part 4, respectively. Where a system command duplicates the action of one of them, this fact will be noted in the description of the system command in this Part.

All system commands can be executed when the terminal is in the execution mode, in which APL operations are executed forthwith upon entry. However, in definition mode, in which sequences of operations are being composed into functions for later execution, commands which call for storing a copy of the workspace, or which might otherwise interfere with the definition process itself, are forbidden. (The two terminal modes are treated more fully in Part 3.)

Classification of commands. System commands are conveniently grouped into five classes with regard to their effect upon the state of the system:

1. Terminal control commands affect the relation of a terminal to the system.

2. Workspace control commands affect the state of the active workspace.

3. Library control commands affect the state of the libraries.

4. Inquiry commands provide information without affecting the state of the system.

5. Communication commands effect the transmission of messages among terminals.

The text that follows is based upon this classification, although it will be seen that certain of the terminal control commands also affect the libraries, and one of the library control commands may sometimes affect the state of the active workspace.

Normal responses and trouble reports. Any entry starting with a right parenthesis will be interpreted by the system as an attempt to execute a system command. When the command is successfully executed, the normal response, if any, will be printed. The expected reponse is given with the description of each command.

If, for any reason, a command cannot be executed, an appropriate trouble report will be printed. The most common report is *INCORRECT COMMAND*. This means that the command was incomplete, mis-spelled, used a wrong modifier, or was otherwise malformed. The corrective action in every case is to enter a properly composed command. The meanings and corrective actions for other trouble reports are given in the notes accompanying the description of each command.

Clear workspace. There are certain transient failures of the system which cause the active workspace to be destroyed. If this should occur, the message *CLEAR WS* will be printed, indicating that the active workspace has been replaced by a clear workspace. (The attributes of a clear workspace are given in the section on workspace control commands, see WC1.) This situation rarely arises, but the probability of its occurrence is slightly higher during the execution of system commands.

Summary. The purposes, forms, responses, and trouble reports for all system commands are summarized in Table 2.1. Where the first word of a command form is more than four characters long, only the first four are significant. The others are included only for mnemonic reasons, and may be dropped or replaced, as desired. For example, )[illegible], )*CLEA*, )*CLEAVER*, etc., are all equivalent.

In general, the elements of a command form must be separated by one (or more) spaces. Spaces are not required immediately following the right parenthesis, or on either side of the colon used with passwords, but can be used without harm.

| Reference and Purpose<br>COMMAND FORM [1,2,3] | NORMAL RESPONSE | TROUBLE REPORTS[4] |
|---|---|---|
| TC1 Sign on designated user and start a work session. | | |
| )NUMBER [KEY] | [TEXT]; PORT,TIME,DATE,USER; SYSTEM; [*SAVED*,TIME,DATE] | 1 2 3 4 5 |
| TC2 End work session. | | |
| )*OFF* [LOCK] | PORT,TIME,DATE,USER CODE; TIME USED | 16 |
| TC3 End work session and hold dial-up connection. | | |
| )*OFF HOLD* [LOCK] | PORT,TIME,DATE,USER CODE; TIME USED | 16 |
| TC4 End work session and store active workspace. | | |
| )*CONTINUE* [LOCK] | [TIME,DATE,*CONTINUE*]; PORT,TIME,DATE,USER CODE; TIME USED | 6 16 |
| TC5 End work session, store active workspace, and hold dial-up connection. | | |
| )*CONTINUE HOLD* [LOCK] | [TIME,DATE,*CONTINUE*]; PORT,TIME,DATE,USER CODE; TIME USED | 6 16 |
| WC1 Activate a clear workspace. | | |
| )*CLEAR* | CLEAR WS | 16 |
| WC2 Activate a copy of a stored workspace. | | |
| )*LOAD* WSID [KEY] | *SAVED*,TIME,DATE | 7 8 16 |
| WC3 Copy a global object from a stored workspace. | | |
| )*COPY* WSID [KEY] NAME | *SAVED*,TIME,DATE | 6 7 8 9 10 16 |
| WC3a Copy all global objects from a stored workspace. | | |
| )*COPY* WSID [KEY] | *SAVED*,TIME,DATE | 6 7 8 10 16 |
| WC4 Copy a global object from a stored workspace, protecting active workspace. | | |
| )*PCOPY* WSID [KEY] NAME | *SAVED*,TIME,DATE; [*NOT COPIED:*,LIST OF OBJECTS] | 6 7 8 9 10 16 |
| WC4a Copy all global objects from a stored workspace, protecting active workspace. | | |
| )*PCOPY* WSID [KEY] | *SAVED*,TIME,DATE; [*NOT COPIED:*,LIST OF OBJECTS] | 6 7 8 10 16 |
| WC5 Gather objects into a group. | | |
| )*GROUP* NAME[S] | NONE | 11 16 |
| WC6 Erase global objects. | | |
| )*ERASE* NAME[S] | [*NOT ERASED:*,LIST OF OBJECTS] | 16 |
| WC7 Set index origin for array operations. | | |
| )*ORIGIN* INTEGER,0-1 | *WAS*,FORMER ORIGIN | 16 |
| WC8 Set maximum for significant digits in output. | | |
| )*DIGITS* INTEGER,1-16 | *WAS*,FORMER MAXIMUM | 16 |
| WC9 Set maximum width for an output line. | | |
| )*WIDTH* INTEGER,30-130 | *WAS*,FORMER WIDTH | 16 |
| WC10 Change workspace identification. | | |
| )*WSID* WSID | *WAS*,FORMER WSID | 16 |
| LC1 Re-store a copy of the active workspace. | | |
| )*SAVE* | TIME,DATE,WSID | 6 12 13 14 16 |
| LC1a Store a copy of the active workspace. | | |
| )*SAVE* WSID [LOCK] | TIME,DATE | 6 12 13 14 16 |
| LC2 Erase a stored workspace. | | |
| )*DROP* WSID | TIME,DATE | 7 14 16 |
| IQ1 List names of defined functions. | | |
| )*FNS* [LETTER] | FUNCTION NAMES | 16 |
| IQ2 List names of global variables. | | |
| )*VARS* [LETTER] | VARIABLE NAMES | 16 |
| IQ3 List names of groups. | | |
| )*GRPS* [LETTER] | GROUP NAMES | 16 |
| IQ4 List membership of designated group. | | |
| )*GRP* NAME | FUNCTION NAMES,VARIABLE NAMES | 16 |
| IQ5 List halted functions (state indicator). | | |
| )*SI* | SEQUENCE OF HALTED FUNCTIONS | 16 |
| IQ6 List halted functions and associated local variables (augmented state indicator). | | |
| )*SIV* | SEQUENCE OF HALTED FUNCTIONS WITH NAMES OF LOCAL VARIABLES | 16 |
| IQ7 Give identification of active workspace. | | |
| )*WSID* | WSID | 16 |
| IQ8 List names of workspaces in designated library. | | |
| )*LIB* [NUMBER] | NAMES OF STORED WORKSPACES | 14 16 |
| IQ9 List ports in use and codes of connected users. | | |
| )*PORTS* | PORT NUMBERS AND ASSOCIATED USER CODES | 16 |
| IQ10 List port numbers associated with designated user code. | | |
| )*PORTS* CODE | PORT NUMBERS | 16 |
| CM1 Address text to designated port. | | |
| )*MSGN* PORT [TEXT] | *SENT* | 15 16 |
| CM2 Address text to designated port, and lock keyboard. | | |
| )*MSG* PORT [TEXT] | *SENT* | 15 16 |
| CM3 Address text to recording terminal (APL Operator). | | |
| )*OPRN* [TEXT] | *SENT* | 15 16 |
| CM4 Address text to recording terminal (APL Operator), and lock keyboard. | | |
| )*OPR* [TEXT] | *SENT* | 15 16 |

| TROUBLE REPORT FORMS | |
|---|---|
| 1 | *NUMBER NOT IN SYSTEM* |
| 2 | *INCORRECT SIGN-ON* |
| 3 | *ALREADY SIGNED ON* |
| 4 | *NUMBER IN USE* |
| 5 | *NUMBER LOCKED OUT* |
| 6 | *NOT WITH OPEN DEFINITION* |
| 7 | *WS NOT FOUND* |
| 8 | *WS LOCKED* |
| 9 | *OBJECT NOT FOUND* |
| 10 | *WS FULL* |
| 11 | *NOT GROUPED, NAME IN USE* |
| 12 | *NOT SAVED, WS QUOTA USED UP* |
| 13 | *NOT SAVED, THIS WS IS* WSID |
| 14 | *IMPROPER LIBRARY REFERENCE* |
| 15 | *MESSAGE LOST* |
| 16 | *INCORRECT COMMAND* |

Notes: 1. Items in brackets are optional.
2. KEY or LOCK: a password set off from preceding text by a colon.
3. WSID: library number and workspace name, or workspace name alone, as required.
4. See insert table of trouble report forms.

Table 2.1: SYSTEM COMMANDS

TERMINAL CONTROL COMMANDS

There is one command for starting a work session, and there are four commands for ending one. The variations in ending allow for automatically storing a copy of the active workspace, and for holding a dial-up telephone connection to the central computer for an immediate start of another work session. The starting command has been described in Part 1 (EC3).

Forced endings. Any action that interrupts a telephone connection for more than five seconds will cause the work session to end, and usually cause a copy of the active workspace to be stored. This provides a safeguard against loss of work in case of failure in the telephone circuits, or accidental loss of power at the terminal. It is also the basis of the disconnect action described in EC4 of Part 1.

A work session can also be stopped remotely, from the system's recording terminal, in an action known as a bounce. As in a disconnect, a copy of the active workspace is usually stored automatically. The bounce may be used when a port is required for a special purpose, or to clear the system of all users before stopping the APL\360 operation completely.

If a work session is ended because of failure of the central computer, the active workspace is not stored.

The *CONTINUE* workspace. When the active workspace is stored automatically, as a result of a disconnect, bounce, or one of the continue commands described below, it goes into the user's private library and is given the name *CONTINUE*. If the active workspace had a password associated with it, *CONTINUE* will be locked with the same password.

If *CONTINUE* is automatically stored, and is not locked, it will be automatically activated at the next sign-on; otherwise, a clear workspace is activated.

Since *CONTINUE* will replace any workspace that may have been previously stored under that name, there is a danger that repeated line failures, while working with a locked workspace, could lead to a complete loss of information. To protect against this possibility, a clear workspace is never stored automatically.

Interrupted activities. An APL operation in progress at the moment of occurrence of a bounce or disconnect may or may not be carried to its normal conclusion. A defined function in progress at such a moment will be suspended, but its progress can be resumed at a later work session in accordance with the procedures given in Part 3. A system command, once begun, will continue to completion regardless of the state of the terminal.

If a bounce or disconnect occurs when the terminal is in definition mode, the definition process is arbitrarily terminated by the system. To proceed with the definition when *CONTINUE* is next activated, the definition mode can be re-established according to the procedures given in Part 3. The continue commands will be rejected in definition mode.

Detailed description. The trouble reports *NO SPACE* and *LIBRARY TABLE FULL* have been omitted from Table 2.1, and are also omitted from the notes below, because their occurence is infrequent, and no corrective action can be taken from a remote terminal. They can arise in response to a continue command or a save command (see section on library control), and signify that certain of the physical resources of the system have been exhausted.

Elapsed time or time of day, given as a system response, is always in hours, minutes, and seconds; two digits for each, separated by periods. A date response is given as month, day and year; two digits for each, separated by slashes. Clock hours are counted continuously from midnight of the indicated day, and if the system runs past midnight it is possible to have time readings well above 24 hours. For example, 34.22.00 07/11/68 would be 22 minutes past 10 AM on July 12, 1968.

| | ACTION | NOTES |
|---|---|---|
| TC1. | Start a work session: This is the sign-on, described in EC3 of Part 1. | See Part 1, EC3. |

TC2. End work session:
Enter )*OFF*
followed by a colon and a password, if desired.

Passwords longer than eight characters are accepted, but only the first eight are meaningful. Spaces around the colon are neutral.

Effect:
1. The currently active workspace will vanish.

There is no effect on any stored workspace.

2. The duration of the work session and the amount of computer time used will be noted internally for later accounting.

3. The password, if used, will become a new lock on the account number.

Once applied, a lock stays in effect until explicitly changed by an ending command that contains a colon.

An existing lock is removed if no password follows the colon.

If a colon is not used, the existing lock, if any, remains in force.

4. A dial-up connection to the central computer will be broken.

Response:
1. The port number, time of day, date, and user code will be printed on one line.

Trouble report:
*INCORRECT COMMAND*

2. Accounting information will be printed on two lines, giving terminal connection time and central computer time.

The time used in this session and cumulative time since the last accounting are given in the standard format, for both terminal time and computer time.

The DATA light on telephone datasets will go out.

TC3. End work session and hold dial-up connection:
Enter )*OFF HOLD*
followed by a colon and a password, if desired.

See note at TC2.

Effect:
1. 2. and 3. Same as for TC2.

4. The dial-up telephone connection will be maintained for 60 seconds, pending a new sign-on.

An attention signal at this time may cause the connection to be broken.

Response:
1. and 2. Same as for TC2.

Trouble report:
*INCORRECT COMMAND*

TC4. End work session and store active workspace:
Enter )*CONTINUE*
followed by a colon and a password, if desired.

See note at TC2.

A bounce has the same effect and response as this command. A disconnect has the same effect, but no response.

Effect:
1. A copy of the currently active workspace will be stored in the user's private library with the name *CONTINUE*. If the workspace had been activated from a stored workspace with a lock, the same lock will be applied to *CONTINUE*.

This effect will not take place if the active workspace is not holding information.

When the workspace is saved it replaces any workspace previously stored with the name *CONTINUE*.

2. 3. and 4. Same as for TC2.

Response:
1. Time of day and date, followed by *CONTINUE*.

This response will be omitted if the workspace was not saved. See note at Effect 1.

2. and 3. Same as for TC2, response 1 and 2.

Trouble reports:
*NOT WITH OPEN DEFINITION* means that the terminal is in definition mode. Close the definition by entering the character ∇. (See mechanics of function definition in Part 3.)

*INCORRECT COMMAND*

See note at TC2.

TC5. End work session, store active workspace, and hold dial-up connection:
Enter )*CONTINUE HOLD* followed by a colon and a password, if desired.

Effect:
1. Same as for TC4.

2. and 3. Same as for TC2.

4. Same as for TC3.

Response:
1. 2. and 3. Same as for TC4.

Trouble reports:
*NOT WITH OPEN DEFINITION*
See TC4.

*INCORRECT COMMAND*

WORKSPACE CONTROL COMMANDS

The commands in this class can replace the active workspace with a clear one, or with a copy of a stored workspace; bring together in the active workspace information from many stored workspaces; form groups within the active workspace; remove unwanted objects from the active workspace; and set controls governing certain operations. No command in this class affects any but the active workspace.

Application packages. The usefulness of a terminal system is enhanced by the availability of many different collections of functions and variables, each of which is organized to satisfy the computational needs of some area of work; for example, standard statistical calculations, exercises for teaching a scholastic subject, complex arithmetic, business accounting, text editing, etc. The workspace-centered organization of APL\360 lends itself to such packaging, because each collection moves as a coherent unit when the workspace containing it is stored or activated.

The copy commands provide a convenient way to assemble packages from components in different workspaces. The group command makes it convenient to have a multiplicity of more specialized packages in a single workspace, sharing common elements, but available individually by copying the appropriate group.

Groups. The group command assigns a single name to a collection of names, in order to provide more convenient reference to selected functions and global variables. The group name can subsequently be used for three purposes: to move a copy of the entire set of referenced objects between workspaces, to incorporate the group members within another group, and to erase, in a single operation, all objects referenced by the group. Each of these is further explained below, in connection with the relevant operation.

Information transfer between workspaces. Information entered or developed within one workspace can be made available within another by means of the copy and protecting-copy commands, which reproduce within the active workspace objects from a stored workspace. These are two

sets of parallel commands which differ only in their treatment of an object in the active workspace which has the same name as an object being reproduced: the copy commands will replace the existing object, whereas the protecting-copy commands will not make the replacement.

A copy command of either type can be applied to an entire workspace or to a single object (i.e., a function, variable, or group). When an entire workspace is copied, all the functions and global variables within it are subject to the operation, but its index origin and output control settings, state indicator, and local variables are left behind.

When a group is copied without protection, both its definition (i.e., the group name and the collection of names composing the group), and the objects referenced by the names within it, are reproduced in the active workspace. When copied with protection, the group itself, or any of the objects referenced by its members, will be omitted in order to protect an object in the active workspace. If the group definition is successfully copied under these circumstances, the names composing it will refer to the global objects by those names in the active workspace, regardless of whether they were copied with the group or present before.

Detailed Description. The term workspace identification is used here to mean either a library number followed by a workspace name, or a workspace name alone. When a name is used alone, the reference is to the user's private library. A key is a colon followed by a password.

| ACTION | NOTES |
|---|---|
| WC1. Activate a clear workspace:<br>Enter *)CLEAR*. | This command is used to make a fresh start, discarding whatever is in the active workspace. |
| Effect:<br>1. A clear workspace will be activated, replacing the presently active workspace. | A clear workspace has no variables, groups, or defined functions.<br><br>Its control settings are: index origin, 1; significant digits, 10; line width, 120.<br><br>Its workspace identification does not match that of any stored workspace. (See section on library control.) |
| Response:<br>1. *CLEAR WS* | Trouble message:<br>*INCORRECT COMMAND* |
| WC2. Activate a copy of a stored workspace:<br>Enter *)LOAD*<br>followed by a space and a workspace identification (with the key, if required). | This command may be used to obtain the use of any workspace in the system whose identification (and password) is known. |
| Effect:<br>1. A copy of the designated workspace will be activated, replacing the presently active workspace. | |
| Response:<br>1. *SAVED*, followed by the time of day and the date that the source workspace was last stored. | Trouble messages:<br>*WS NOT FOUND*<br>means there is no stored workspace with the given identification.<br><br>*WS LOCKED*<br>means that no key, or the wrong key, was used when one was required.<br><br>*INCORRECT COMMAND* |
| WC3. Copy a global object from a stored workspace:<br>Enter *)COPY*<br>followed by a space and a workspace identification (with the key, if required), followed by a space and the name of the object to be copied. | A global object may be a group, function, or global variable.<br><br>When applied to a group, all copy commands operate both on the group definition and on objects referenced by the group members. |
| Effect:<br>1. A copy of the designated object will appear in the active workspace with global significance, replacing existing global homonyms. | Members of a group do not necessarily have referents; but a group member without a referent in the source workspace may find one in the active workspace. |

Response:
1. *SAVED*, followed by the time of day and the date that the source workspace was last stored.

Trouble messages:
*NOT WITH OPEN DEFINITION*
means that the terminal is in definition mode. Either close the definition by entering ∇, or defer the copy operation.

*WS NOT FOUND*
See WC2.

*WS LOCKED*
See WC2.

*OBJECT NOT FOUND*
means that the designated workspace does not contain a global object with the given name.

*WS FULL*
means that the active workspace could not contain all the material requested: if copied at all, a variable will be copied completely; a partially copied function will leave the terminal in definition mode; some objects may be completely overlooked. Status may be determined by using appropriate inquiry commands.

*INCORRECT COMMAND*

WC3a. Copy all global objects from a stored workspace:
Enter )*COPY*
followed by a space and a workspace identification (with the key, if required).

See notes at WC3.

Effect:
1. A copy of all functions, groups, and global variables in the source workspace will appear in the active workspace with global significance, replacing existing global homonyms.

Local variables, the state indicator, and settings for origin, significant digits, and width are not copied.

Response:
1. *SAVED*, followed by the time of day and the date that the source workspace was last stored.

Trouble messages:
*NOT WITH OPEN DEFINITION*
*WS NOT FOUND*
*WS LOCKED*
*WS FULL*
*INCORRECT COMMAND*
See WC3 for all meanings.

WC4. Copy a global object from a stored workspace, protecting the active workspace:
Enter )*PCOPY*
followed by a space and a workspace identification (with the key, if required), followed by a space and the name of the object to be copied.

See notes at WC3.

Effect:
1. A copy of the designated object will appear in the active workspace unless there is an existing global homonym.

When a group definition is copied, any member whose referent was blocked will, perforce, refer to the referent of its homonym.

Response:
1. *SAVED*, followed by the time of day and the date that the source workspace was last stored.

2. *NOT COPIED*:, followed by the names of objects not copied, will be printed if appropriate.

Trouble messages:
*NOT WITH OPEN DEFINITION*
*WS NOT FOUND*
*WS LOCKED*
*OBJECT NOT FOUND*
*WS FULL*
*INCORRECT COMMAND*
See WC3 for all meanings.

**WC4a. Copy all global objects from a stored workspace, protecting the active workspace:**
Enter *)PCOPY*
followed by a space and a workspace identification (with the key, if required).

See notes at WC3.

**Effect:**
1. A copy of all global objects in the source workspace which do not have global homonyms in the active workspace will appear in the active workspace.

See note at WC3a, Effect 1.

See note at WC4, Effect 1.

**Response:**
1. *SAVED*, followed by the time of day and the date that the source workspace was last stored.

2. *NOT COPIED:*, followed by the names of objects not copied, will be printed if appropriate.

**Trouble messages:**
*NOT WITH OPEN DEFINITION*
*WS NOT FOUND*
*WS LOCKED*
*WS FULL*
*INCORRECT COMMAND*
See WC3 for all meanings.

**WC5. Gather names into a group:**
Enter *)GROUP*
followed by a space and one or more names separated by spaces.

**Effect:**
1. The first name will be the name of a group having the other names as members, subject to the rules given in the adjacent notes. An existing group with the same name will be superseded.

The first name used in the command must not be the name of a function or global variable.

Any name may be a member of a group; names of groups, functions, and global variables, and names without current global referents are all acceptable.

Members may be added to an existing group by using the group name twice in the command: as the first name and as another.

2. If only one name is used in the command, no group is formed, and an existing group by that name is dispersed.

When a group is dispersed the group definition is destroyed, but the referents of the group members are unaffected.

**Response:** None.

**Trouble reports:**
*NOT GROUPED, NAME IN USE*
means that the first name used in the command is the name of a function or global variable. Erase the offending object, or use a different name.

*INCORRECT COMMAND*

WC6. Erase global objects:
Enter )*ERASE*
followed by a space and the names of objects to be deleted, separated by spaces.

This is the only way to remove a global variable, and the most convenient way to remove a collection of objects.

Effect:
1. Named objects having global significance, other than pendent functions, will be expunged.

Names which do not refer to global objects are ignored.

When a group is erased, both the group and the referents of its members are expunged.

Response:
1. *ERASED:*, followed by the names of functions not erased will be printed, if appropriate.

Trouble report:
*INCORRECT COMMAND*

WC7. Set index origin for array operations: Enter the characters )*ORIGIN*
followed by a space and a 0 or 1.

A dynamically executable equivalent function is available (see Part 4).

Effect:
1. First elements of arrays in the workspace will be numbered zero or one, as indicated, and subsequent use of index-dependent APL operations will be appropriately affected.

These matters are explained in Part 3.

Response:
1. *WAS*, followed by the former origin.

Trouble report:
*INCORRECT COMMAND*

WC8. Set maximum for significant digits in output:
Enter )*DIGITS*
followed by a space and an integer between 1 and 16 inclusive.

A dynamically executable equivalent function is available (see Part 4).

Effect:
1. Subsequent output of numbers will show no greater number of significant digits than indicated.

This command has no effect on the precision of internal calculations, which is approximately 16 decimal digits.

Response:
1. *WAS*, followed by the former maximum.

Trouble report:
*INCORRECT COMMAND*

WC9. Set maximum width for an output line:
Enter )*WIDTH*
followed by a space and an integer between 30 and 130 inclusive.

A dynamically executable equivalent function is available (see Part 4).

Effect:
1. Subsequent output of all kinds, except messages between terminals, will be limited to a line width no greater than the number of spaces indicated.

This affects neither the mechanical margin stops nor the allowable length of input lines.

Response:
1. *WAS*, followed by the former maximum width.

Trouble report:
*INCORRECT COMMAND*

WC10. Change workspace identification:
Enter *)WSID*
followed by a space and a workspace identification.

This command can be used to guard against inadvertently changing a stored workspace that has just been loaded; and conversely, to enable the replacement of a stored workspace without first using the drop command, when the active workspace came from a different source. (See section on library control commands.)

Effect:
1. The active workspace will assume the specified identification. A lock associated with the workspace will be retained.

See command LC1 for the implications of this.

Response:
1. *WAS*, followed by the former workspace identification.

Trouble report:
*INCORRECT COMMAND*

LIBRARY CONTROL COMMANDS

There are two basic operations performed by the commands in this class. The save commands cause a copy of an active workspace to be stored in a library, and the drop command causes such a stored copy to be destroyed.

The save commands and the load command are symmetric, in the sense that a load command destroys an active workspace by replacing it with a copy of a stored workspace, while a save command may destroy a stored workspace by replacing it with a copy of the active workspace.

Continuity of work. When a workspace is stored, an exact copy of the active workspace is made, including the state indicator and intermediate results from the partial execution of halted functions. These functions can be restarted without loss of continuity (see Part 3), which permits considerable flexibility in planning use of the system. For example, lengthy calculations do not have to be completed at one terminal session; student work can be conducted over a series of short work periods, to suit class schedules; and mathematical experimentation or the exploration of system models can be done over long periods of time, at the investigator's convenience.

Workspace identification. A library number and a name, together, uniquely identify each stored workspace in the system. An active workspace is also identified by a library number and a name, and as copies of stored workspaces are activated, or copies of the active workspace are stored, the identification of the active workspace may change according to the following rules:

1. A workspace activated from a library assumes the identification of its source.

2. When a copy of the active workspace is stored, the active workspace assumes the identification assigned to the stored copy.

3. The library number and name may be arbitrarily changed by the use of command WC10.

4. A clear workspace activated by a clear command, a sign-on, or a system failure is called *CLEAR WS*, which cannot be the name of a stored workspace.

The identification of active workspaces is used in two ways. First, as a safeguard against the inadvertent replacement of a stored workspace by an unrelated one: an attempt to replace, by a copy of the active workspace, any stored workspace other than the one with the same identification (or the one named *CONTINUE*), will be stopped. Second, as a convenience when the active workspace is to be re-stored with changes: the use of the command *)SAVE*, without modification, implicitly uses the identification of the active workspace.

Library and account numbers. A user's account number is also the number of his private library. The numbers of public libraries range from 1 to 999, and do not correspond to any account number.

Each stored workspace has implicitly associated with it the account number signed on at the terminal from which the save command was entered, and may not be either replaced or erased, except from a terminal signed on with the same account number. Thus, a user is prevented from affecting

the state of another user's private library, or tampering with public library workspaces which he did not store. He may, of course, activate a copy of any workspace stored in the system, if he knows the library number and name (and password, if required).

Storage allotment. A user of APL\360 is assigned library space in terms of the maximum number of stored workspaces he may have at one time. This quota applies to the combined total of workspaces stored either in his private library or in public libraries. The allotment for each user is determined by those responsible for the general management of a particular system, and can be changed from the recording terminal, as required, within the bounds of the physical resources of the system.

Up to the number in his quota, a user may assign arbitrary names to the workspaces he stores. Beyond that point he always has available one workspace named *CONTINUE* in his private library.

Use of the *CONTINUE* workspace. This workspace has the property that it may be freely replaced by an active workspace having any identification whatsoever. It is thus always available as temporary storage, but carries with it the danger of being easily replaced, as described in the section on terminal control commands.

The attributes of the *CONTINUE* workspace are the same whether stored as a result of a continue command, disconnect, or bounce, or stored by virtue of a save command using that name. In the last case, the active workspace assumes the name *CONTINUE*, as it would any other name under like circumstances.

Purging a workspace. The sequence of commands, *)SAVE ABC123*, *)CLEAR*, *)COPY ABC123*, will purge the active workspace, clearing it of all but its functions, groups, and global variables, and reset its controls (see WC1). This often results in more usable space than can otherwise be realized. Subsequently, the commands *)WSID ABC123* and *)SAVE* may be used to store a copy of the purged workspace under its former name.

Detailed Description. The term workspace identification will be used with the same significance as for the workspace control commands.

| ACTION | NOTES |
|---|---|
| **LC1. Re-store a copy of the active workspace:**<br>Enter *)SAVE* | |
| Effect:<br>1. A copy of the active workspace will replace the stored workspace with the same identification. | New workspaces can be stored by this command only if the identification of the active workspace has been changed by WC10. |
| 2. A password associated with the active workspace will continue in effect, and the stored workspace will be locked with this password. | This forestalls inadvertent omission of a lock while actively engaged with a confidential workspace. |
| Response:<br>1. The time of day, date, and workspace identification will be printed. | Trouble reports:<br>*NOT WITH OPEN DEFINE N*<br>means that the terminal is in function definition mode. Either close the definition by entering ∇, or defer the save operation.<br><br>*NOT SAVED, WS QUOTA SET*<br>means that the allotted number of stored workspaces has previously been reached. Unless this is increased, the workspace can be stored only by replacing a workspace already stored. *CONTINUE* may be replaced directly; any other must be erased first, or the identification of the active workspace must be made to match by WC10. |

*NOT SAVED, THIS WS IS* [illegible]

results from the fact that *CLEAR WS* cannot be the name of a stored workspace. Either change the name by WC10, or use LC1a.

*IMPROPER LIBRARY REFERENCE*
means that an attempt was made either to replace a stored workspace that is not under control of the account number signed on at the terminal, or to store into a non-existent library.

*INCORRECT COMMAND*

**LC1a. Store a copy of the active workspace:**
Enter *)SAVE*
followed by a space and a workspace identification, with a colon and password, if desired.

This form of the save command allows new workspaces to be added to a library more conveniently, and permits locks to be added or removed from workspaces already present.

**Effect:**
1. A copy of the active workspace will be stored with the designated identification, and with the assigned lock, if a password was used.

A stored workspace with the same identification will be replaced.

A lock on a stored workspace will not be retained if the command does not include a lock explicitly.

2. The active workspace will assume the workspace identification used in the command.

To this extent only, this command may affect the state of the active workspace.

**Response:**
1. The time of day and date will be printed.

**Trouble reports:**
*NOT WITH OPEN DEFINITION*
means the same as for LC1.

*NOT SAVED, WS QUOTA USED UP*
means the same as for LC1.

*NOT SAVED, THIS WS IS*
followed by identification of the active workspace, means a stored workspace with the identification used in the command exists, but this identification does not match that of the active workspace.

*IMPROPER LIBRARY REFERENCE*
means the same as for LC1.

*INCORRECT COMMAND*

**LC2. Erase a stored workspace:**
Enter *)DROP*
followed by a space and a workspace identification.

Since a key is not used, a locked workspace whose key has been lost can always be removed from the system.

**Effect:**
1. The designated stored workspace will be expunged.

This command has no effect on the active workspace, regardless of its identification.

**Response:**
1. The time of day and date will be printed.

**Trouble reports:**
*IMPROPER LIBRARY REFERENCE*
means that an attempt was made to drop a workspace stored by another user.

*WS NOT FOUND*
means that there is no stored workspace with the identification used in the command.

*INCORRECT COMMAND*

INQUIRY COMMANDS

Most of the commands in this class concern the state of the active workspace. Of the others, one command lists the names of workspaces in libraries, and two commands are useful for locating another user at a connected terminal, in order to communicate with him.

User codes. The communication commands described in the next section require that the port number of the person to be addressed be known. The inquiry commands that provide this information operate through the device of user codes, which serve within the system as partial identification of users. (The user account numbers, which completely identify users within the system, are not used for this purpose, and are treated as private information.) A user code comprises the first three characters of his name, as it appears in the sign-on response (Part 1, EC3, Response 2).

A user code is considered to be only partial identification because it may not be unique. Therefore, these commands should be used advisedly: before addressing substantive messages to a terminal which has been identified by a user code, further confirmation of the receiver's identity should be sought.

Detailed description.

| ACTION | NOTES |
|---|---|
| IQ1. List names of defined functions:<br>Enter *)FNS*<br>followed by an alphabetic character, if desired.<br><br>Effect: None.<br><br>Response:<br>1. The names of defined functions in the active workspace will be printed alphabetically, starting with the specified letter. If a letter was not used, all function names will be listed. | Trouble message:<br>*INCORRECT COMMAND* |
| IQ2. List names of global variables:<br>Enter *)VARS*<br>followed by an alphabetic character, if desired.<br><br>Effect: None.<br><br>Response:<br>1. The names of global variables in the active workspace will be printed alphabetically, starting with the specified letter. If a letter was not used, all names of global variables will be listed. | Trouble message:<br>*INCORRECT COMMAND* |
| IQ3. List names of groups:<br>Enter *)GRPS*<br>followed by an alphabetic character, if desired.<br><br>Effect: None.<br><br>Response:<br>1. The names of groups in the active workspace will be printed alphabetically, starting with the letter used. If a letter was not used, all group names will be listed. | Trouble message:<br>*INCORRECT COMMAND* |

**IQ4. List membership of designated group:**
Enter *)GRP*
followed by the name of the group.

**Effect:** None.

**Response:**
1. The names in the group will be printed.

There will be no response if there is no group with the designated name in the active workspace.

**Trouble message:**
*INCORRECT COMMAND*

**IQ5. List halted functions:**
Enter *)SI*

**Effect:** None.

**Response:**
1. The names of halted functions will be listed, most recent ones first. With each name will be given the line number on which execution stopped. Suspended functions will be distinguished from pendent functions by an asterisk.

The line numbers on which halted functions have stopped are available for dynamic use through the system-dependent functions ⌶26 and ⌶27. (See Part 3.)

This display is the *state indicator*; its significance and use is explained in Part 3.

**Trouble message:**
*INCORRECT COMMAND*

**IQ6. List halted functions with names of local variables:**
Enter *)SIV*

**Effect:** None.

**Response:**
1. The response will be the same as for IQ5, except that with each function listed there will appear a listing of its local variables.

**Trouble message:**
*INCORRECT COMMAND*

**IQ7. Give identification of active workspace:**
Enter *)WSID*

**Effect:** None.

**Response:**
1. The identification of the active workspace will be printed. The library number will be included only if it differs from the account number associated with the terminal.

**Trouble message:**
*INCORRECT COMMAND*

**IQ8. List names of stored workspaces:**
Enter *)LIB*
followed, if necessary, by a library number.

A library number is not required for listings of the user's private library.

**Effect:** None.

**Response:**
1. The names of workspaces in the designated library will be printed. If no number was used, the account number associated with the terminal will be taken as the library number.

**Trouble messages:**
*IMPROPER LIBRARY REFERENCE*
means that an attempt was made to obtain a listing of another user's private library, or of a non-existent library.

*INCORRECT COMMAND*

IQ9. List ports in use and codes of connected users:
Enter *)PORTS*

Effect: None.

Response:
1. Port numbers in use will be printed with the associated user code.

Trouble message:
*INCORRECT COMMAND*

IQ10. List port numbers associated with designated user code:
Enter *)PORTS*
followed by the user code.

User codes are not necessarily unique, and the information derived from this command and IQ9 should be used advisedly.

Effect: None.

Response:
1. The port numbers of connected users identified by the code will be printed.

Trouble message:
*INCORRECT COMMAND*

COMMUNICATION COMMANDS

There are two pairs of commands in this class. One pair addresses any connected terminal, and one pair addresses only the system recording terminal.

A message can be received by a terminal only when its keyboard is locked, and except for public address announcements from the system recording terminal, only if it is also not in the process of function execution. Hence, to facilitate two-way communication, one of each pair of communication commands results in locking the keyboard of the sending terminal, pending the receipt of a reply. A keyboard so locked can be unlocked by an attention signal.

Incoming messages from the system recording terminal are prefixed by 'PT ', and public address messages are prefixed by 'A':.

If the interaction at a terminal must be interrupted for a prolonged period while the terminal is still connected, it is good practice to lock the keyboard so that a message may be received. This can be done by addressing a message of the proper type to the terminal's own port number.

Detailed description. The length of a message is restricted to a single line, not exceeding 120 characters in length. However, messages are not subject to the width settings of either the sending or receiving terminal.

| ACTION | NOTES |
|---|---|
| CM1. Address text to designated port:<br>Enter *)MSGN*<br>followed by a port number and any one-line text. | A message addressed to an unused or non-existent port will be reflected back to the sending terminal, which then plays the role of both sender and receiver. |

Effect:
1. The keyboard will lock while the text is being transmitted.

2. The text will be printed at the receiving terminal, prefixed by the port number of the sending terminal.

3. The keyboard will unlock when the transmission is completed.

Response:
1. *SENT*

Trouble message:
*MESSAGE LOST*
means just that. It happens when attention is signalled before a message is delivered, or an equivalent transmission disturbance occurs.

*INCORRECT COMMAND*

**CM2. Address text to designated port and lock keyboard:**
Enter *)MSG*
followed by a port number and any one-line text.

**Effect:**
1. Same as CM1 effect 1.

2. Same as CM1, Effect 2, except for a prefix *R*, to indicate that a reply is awaited.

3. The keyboard will remain locked after the response is printed.

**Response:**
1. *SENT*

See note at CM1.

The keyboard can be unlocked, before receiving a reply, by means of an attention signal.

**Trouble message:**
*MESSAGE LOST*
See CM1.

*INCORRECT COMMAND*

**CM3. Address text to system recording terminal:**
Enter *)OPRN*
followed by any one-line text.

**Effect:**
1. 2. and 3. Same as CM1.

**Response:**
1. *SENT*

See note at CM1.

**Trouble message:**
*MESSAGE LOST*
See CM1.

*INCORRECT COMMAND*

**CM4. Address text to system recording terminal and lock keyboard:**
Enter *)OPR*
followed by any one-line text.

**Effect:**
1. 2. and 3. Same as CM2.

**Response:**
1. *SENT*

See note at CM1.

**Trouble message:**
*MESSAGE LOST*
See CM1.

*INCORRECT COMMAND*

PART 3

## THE LANGUAGE

The APL\360 Terminal System executes system commands or mathematical statements entered on a terminal typewriter. The system commands were treated in Part 2; the mathematical statements will be treated here.

Acceptable statements may employ either primitive functions (e.g. + - × ÷) which are provided by the system, or defined functions, which the user provides by entering their definitions on the terminal.

If system commands are not used, the worst that can possibly result from erroneous use of the keyboard is the printing of an error report. It is therefore advantageous to experiment freely and to use the system itself for settling any doubts about its behavior. For example, to find what happens in an attemped division by zero, simply enter the expression 4÷0. If ever the system seems unusually slow to respond, execute an attention signal to interrupt execution and unlock the keyboard.

The Sample Terminal Session of Appendix A shows actual intercourse with the system which may be used as a model in gaining facility with the terminal. The examples follow the text and may well be studied concurrently. More advanced programming examples appear in Appendix B.

The primitive functions and the defined functions available in libraries can be used without knowledge of the means of defining functions. These means are treated in the four contiguous sections beginning with Defined Functions and ending with Homonyms. These sections may be skipped without loss of continuity.

### FUNDAMENTALS

Statements. Statements are of two main types, the branch (denoted by → and treated in the section on Defined Functions), and the specification. A typical specification statement is of the form

```
X←...
```

This statement assigns to the variable *X* the value resulting from the expression to the right of the specification arrow. If the variable name and arrow are omitted, the resulting value is printed. For example:

```
      3×4
12
```

Results typed by the system begin at the left margin whereas entries from the keyboard are automatically indented. The keyboard arrangement is shown in Figure 1.2.

Scalar and vector constants. All numbers entered via the keyboard or typed out by the system are in decimal, either in conventional form (including a decimal point if appropriate) or in exponential form. The exponential form consists of an integer or decimal fraction followed immediately by the symbol *E* followed immediately by an integer. The integer following the *E* specifies the power of ten by which the part preceding the *E* is to be multiplied. Thus 1.44*E*2 is equivalent to 144.

Negative numbers are represented by a negative sign immediately preceding the number, e.g., ¯1.44 and ¯144*E*¯2 are equivalent negative numbers. The negative sign can be used only as part of a constant and is to be distinguished from the negation function which is denoted, as usual, by the minus sign -.

A constant vector is entered by typing the constant components in order, separated by one or more spaces. A character constant is entered by typing the character between quotation marks, and a sequence of characters entered in quotes represents a vector whose successive components are the characters themselves. Such a vector is printed by the system as the sequence of characters, with no enclosing quotes and with no separation of the successive elements. The quote character itself must be typed in as a pair of quotes. Thus, the abbreviation of *CAN'T* is entered as *'CAN''T'* and prints as *CAN'T*.

Names and Spaces. As noted in Part 2, the name of a variable or defined function may be any sequence of letters or digits beginning with a letter and not containing a space. A letter may be any of the characters *A* to *Z*, or any one of these characters underscored, e.g., *A* or *Z*.

Spaces are not required between primitive functions and constants or variables, or between a succession of primitive functions, but they may be used if desired. Spaces are needed to separate names of adjacent defined functions, constants, and variables. For example, the expression +/ may be entered with no spaces, but if *F* is a defined

function, then the expression `3 F 4` must be entered with the indicated spaces. The exact number of spaces used in succession is of no importance and extra spaces may be used freely.

Overstriking and erasure. Backspacing serves only to position the carriage and does not cause erasure or deletion of characters. It can be used:

1. to insert missing characters (such as parentheses) if space has previously been left for them,

2. to form compound characters by overstriking (e.g. ¢ and ! ), and

3. to position the carriage for erasure, which is effected by striking the linefeed (marked ATTN on IBM 2741 terminals). The linefeed has the effect of erasing the character at the position of the carriage, and all characters to the right.

End of Statement. The end of a statement is indicated by striking the carriage return (followed, on some terminals, by an explicit transmission signal as described in Part 1). The typed entry is then interpreted exactly as it appears on the page, regardless of the time sequence in which the characters were typed.

Order of execution. In a compound expression such as `3×4+6÷2`, the functions are executed (evaluated) from rightmost to leftmost, regardless of the particular functions appearing in the expression. (The foregoing expression evaluates to 21.) When parentheses are used, as in the expression `W←(3⌈Q)÷X×Y-Z`, the same rule applies, but, as usual, an enclosed expression must be completely evaluated before its results can be used. Thus, the foregoing expression is equivalent to `W←(3⌈Q)÷(X×(Y-Z))`.

In general, the rule can be expressed as follows: every function takes as its righthand argument the entire expression to its right, up to the right parenthesis of the pair that encloses it.

Error reports. The attempt to execute an invalid statement will cause one of the error reports of Table 3.1 to be typed out. The error report will be followed by the offending statement with a caret typed under the point in the statement where the error was detected. If the caret lies to the right of a specification arrow, the specification has not yet been performed.

| TYPE | Cause; CORRECTIVE ACTION |
|---|---|
| *CHARACTER* | Illegitimate overstrike. |
| *DEPTH* | Excessive depth of function execution. CLEAR STATE INDICATOR. |
| *DOMAIN* | Arguments not in the domain of the function. |
| *DEFN* | Misuse of ∇ or □ symbols:<br>1. ∇ is in some position other than the first.<br>2. The function is pendent. DISPLAY STATE INDICATOR AND CLEAR AS REQUIRED.<br>3. Use of other than the function name alone in reopening a definition.<br>4. Improper request for a line edit or display. |
| *INDEX* | Index value out of range. |
| *LABEL* | Name of already defined function used as a label, or colon used other than in function definition and between label and statement. |
| *LENGTH* | Shapes not conformable. |
| *RANK* | Ranks not conformable. |
| *RESEND* | Transmission failure. RE-ENTER. IF CHRONIC, REDIAL OR HAVE TERMINAL OR PHONE REPAIRED. |
| *SYNTAX* | Invalid syntax; e.g., two variables juxtaposed; function used without appropriate arguments as dictated by its header; unmatched parentheses. |
| *SYMBOL TABLE FULL* | Too many names used. ERASE SOME FUNCTIONS OR VARIABLES, THEN SAVE, CLEAR, AND COPY. |
| *SYSTEM* | Fault in internal operation of APL\ 360. RELOAD OR SAVE, CLEAR, AND COPY. SEND TYPED RECORD, INCLUDING ALL WORK LEADING TO THE ERROR, TO THE SYSTEM MANAGER. |
| *VALUE* | Use of name which has not been assigned 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). CLEAR STATE INDICATOR, ERASE NEEDLESS OBJECTS, OR REVISE CALCULATIONS TO USE LESS SPACE. |

Table 3.1 ERROR REPORTS

If an invalid statement is encountered during execution of a defined function, the error report includes the function name and the line number of the invalid statement. The recommended procedure at this point is to enter a right arrow (→) alone, and then retry with an amended statement. The matter is treated more fully in the section on Suspended Function Execution.

Names of primitive functions. The primitive functions of the language are summarized in Tables 3.2 and 3.8, and will be discussed individually in subsequent sections. The tables show one suggested name for each function. This is not intended to discourage the common mathematical practice of vocalizing a function in a variety of ways (for example, $X \div Y$ may be expressed as "$X$ divided by $Y$", or "$X$ over $Y$"). Thus, the expression $\rho M$ yields the dimension of the array $M$, but the terms size or shape may be preferred both for their brevity and for the fact that they avoid potential confusion with the dimensionality or rank of the array.

The importance of such names and synonyms diminishes with familiarity. The usual tendency is toward the use of the name of the symbol itself (e.g., "rho" (ρ) for "size", and "iota" (⍳) for "index generator"), probably to avoid unwanted connotations of any of the chosen names.

SCALAR FUNCTIONS

Each of the primitive functions is classified as either scalar or mixed. Scalar functions are defined on scalar (i.e., individual) arguments and are extended to arrays in four ways: element-by-element, reduction, inner product, and outer product, as described in the section on Functions on Arrays. Mixed functions are discussed in a later section.

The scalar functions are summarized in Table 3.2. Each is defined on real numbers or, as in the case of the logical functions and and or, on some subset of them. No functional distinction is made between "fixed point" and "floating point" numbers, this being primarily a matter of the representation in a particular medium, and the user of the terminal system need have no concern with such questions unless his work strains the capacity of the machine with respect to either space or accuracy.

Three different representations for numbers are used internally, and transformations among them are carried out automatically. Integers less than 2 to the power 52 are carried with full precision; larger numbers and non-integers are carried to a precision of about 16 decimal digits.

| Monadic form $fB$ | | f | Dyadic form $AfB$ | |
|---|---|---|---|---|
| Definition or example | Name | | Name | Definition or example |
| $+B \leftrightarrow 0+B$ | Plus | + | Plus | $2+3 \leftrightarrow 5$ |
| $-B \leftrightarrow 0-B$ | Negative | - | Minus | $2-3.2 \leftrightarrow ¯1.2$ |
| $\times B \leftrightarrow (B>0)-(B<0)$ | Signum | × | Times | $2\times3.2 \leftrightarrow 6.4$ |
| $\div B \leftrightarrow 1\div B$ | Reciprocal | ÷ | Divide | $2\div3.2 \leftrightarrow 0.625$ |
| $B$ \| $\lceil B$ \| $\lfloor B$<br>3.1 \| 4 \| 3<br>¯3.1 \| ¯3 \| ¯4 | Ceiling<br>Floor | ⌈ | Maximum | $3\lceil7 \leftrightarrow 7$ |
| | | ⌊ | Minimum | $3\lfloor7 \leftrightarrow 3$ |
| $\star B \leftrightarrow (2.71828..)\star B$ | Exponential | ⋆ | Power | $2\star3 \leftrightarrow 8$ |
| $\bullet\star N \leftrightarrow N \leftrightarrow \star\bullet N$ | Natural logarithm | ⍟ | Logarithm | $A\bullet B \leftrightarrow$ Log $B$ base $A$<br>$A\bullet B \leftrightarrow (\bullet B)\div\bullet A$ |
| $\lvert ¯3\ 14 \leftrightarrow 3\ 14$ | Magnitude | \| | Residue | Case \| $A\mid B$<br>$A\neq0$ \| $B-A\times B\div A$<br>$A=0, B\neq0$ \| 0<br>$A=0, B<0$ \| Domain error |
| $!3 \leftrightarrow 6$<br>$!B \leftrightarrow B\times!B-1$<br>or $!B \leftrightarrow$ Gamma$(B+1)$ | Factorial | ! | Binomial coefficient | $A!B \leftrightarrow (!B)\div(!A)\times!B-A$<br>$2!5 \leftrightarrow 10$ $3!5 \leftrightarrow 10$ |
| $?B \leftrightarrow$ Random choice from $\iota B$ | Roll | ? | Deal | A Mixed Function (See Table 3.8) |
| $\circ B \leftrightarrow 3.14159..\times B$ | Pi times | ○ | Circular | See Table at left |
| $\sim1 \leftrightarrow 0$ $\sim0 \leftrightarrow1$ | Not | ~ | | |
| | | ∧ | And | A \| B \| A∧B<br>0 \| 0 \| 0<br>0 \| 1 \| 0<br>1 \| 0 \| 0<br>1 \| 1 \| 1 |
| | | ∨ | Or | |
| | | ⍲ | Nand | |
| | | ⍱ | Nor | |
| | | < | Less | Relations<br>Result is if the relation holds, if it does not: |
| | | ≤ | Not greater | |
| | | = | Equal | |
| | | ≥ | Not less | |
| | | > | Greater | |
| | | ≠ | Not Equal | |

| $(-A)\circ B$ | $A$ | $A\circ B$ |
|---|---|---|
| $\sqrt{1-B\star2})\star5$ | 0 | $(1-B\star2)\star.5$ |
| Arcsin $B$ | 1 | Sine $B$ |
| Arccos $B$ | 2 | Cosine $B$ |
| Arctan $B$ | 3 | Tangent $B$ |
| $\sqrt{1+B\star2})\star5$ | 4 | $(1+B\star2)\star.5$ |
| Arcsinh $B$ | 5 | Sinh $B$ |
| Arccosh $B$ | 6 | Cosh $B$ |
| Arctanh $B$ | 7 | Tanh $B$ |

Table of Dyadic ○ Functions

Table 3.2: PRIMITIVE SCALAR FUNCTIONS

For operations such as floor and ceiling, and in comparisons, a "fuzz" of about $1E^{-}13$ is applied in order to avoid anomalous results that might otherwise be engendered by doing decimal arithmetic on a binary machine.

Two of the functions of Table 3.2, the relations $\neq$ and $=$, are defined on characters as well as on numbers.

Monadic and dyadic functions. Each of the functions defined in Table 3.2 may be used in the same manner as the familiar arithmetic functions + - × and ÷. Most of the symbols employed may denote either a monadic function (which takes one argument) or a dyadic function (which takes two arguments). For example, $\lceil Y$ denotes the monadic function ceiling applied to the single argument $Y$, and $X \lceil Y$ denotes the dyadic function maximum applied to the two arguments $X$ and $Y$. Any such symbol always denotes a dyadic function if possible, i.e., it will take a left argument if one is present.

At this point it may be helpful to scrutinize each of the functions of Table 3.2 and to work out some examples of each, either by hand or on a terminal. However, it is not essential to grasp all of the more advanced mathematical functions (such as the hyperbolic functions sinh, cosh, and tanh, or the extension of the factorial to non-integer arguments) in order to proceed. Treatments of these functions are readily available in standard texts.

Certain of the scalar functions deserve brief comment. The residue function $A \mid B$ has the usual definition of residue used in number theory. For positive integer arguments this is equivalent to the remainder obtained on dividing $B$ by $A$, and may be stated more generally as the smallest non-negative member of the set $B-N \times A$, where $N$ is any integer.

This formulation covers the case of a zero left argument as shown in Table 3.2. The conventional definition is extended in two further respects:

1. The left argument $A$ need not be positive; however, the value of the result depends only on the magnitude of $A$.

2. The arguments need not be integral. For example, $1 \mid 2.6$ is 0.6 and $1.5 \mid 8$ is 0.5.

The expression $^{-}8 * .5$ (square root of $^{-}8$) yields a domain error, but $^{-}8 * 1 \div 3$ has the value $^{-}2$. More generally, $A * B$ is valid for $A<0$ if the right argument is (a close approximation to) a rational number with an odd denominator not greater than 85.



The factorial function $!N$ is defined in the usual way as the product of the first $N$ positive integers. It is also extended to non-integer values of the argument $N$ and is equivalent to the Gamma function of $N+1$.

The function $A!B$ (pronounced $A$ out of $B$) is defined as $(!B) \div (!A) \times !B-A$. For integer values of $A$ and $B$, this is the number of combinations of $B$ things taken $A$ at a time. (It is related to the Complete Beta function as follows: $\text{Beta}(P,Q) \leftrightarrow \div Q \times (P-1)!P+Q-1$.)

The symbols $< \leq = \geq > \neq$ denote the relations less than, less than or equal, etc., in the usual manner. However, an expression of the form $A<B$ is treated not as an assertion, but as a function which yields a 1 if the proposition is true, and 0 if it is false. For example:

```
      3≤7
1
      7≤3
0
```

When applied to logical arguments (i.e., arguments whose values are limited to 0 and 1), the six relations are equivalent to six of the logical functions of two arguments. For example, $\leq$ is equivalent to material implication, and $\neq$ is equivalent to exclusive-or. These six functions together with the and, or, nand, and nor shown in Table 3.2 exhaust the nontrivial logical functions of two logical arguments.

Vectors. Each of the monadic functions of Table 3.2 applies to a vector, element by element. Each of the dyadic functions applies element by element to a pair of vectors of equal dimension or to one scalar and a vector of any dimension, the scalar being used with each component of the vector. For example:

```
      1 2 3 4×4 3 2 1
4  6  6  4
      2+1 2 3 4
3  4  5  6
      1 2 3 4⌈2
2  2  3  4
```

Index generator. If $N$ is a non-negative integer, then $\iota N$ denotes a vector of the first $N$ integers. The dimension of the vector $\iota N$ is therefore $N$; in particular, $\iota 1$ is a vector of length one which has the value 1, and $\iota 0$ is a vector of

dimension zero, also called an empty vector. The empty vector prints as a blank. For example:

```
      ⍳4
1 2 3 4
      ⍳5
1 2 3 4 5
      ⍳0

      6-⍳6
5 4 3 2 1 0
      2×⍳0

      2×⍳6
2 4 6 8 10 12
```

Empty vector prints as a blank

Scalar applies to all (i.e., 0) elements of ⍳0, resulting in an empty vector

The index generator is one of the class of mixed functions to be treated in detail later; it is included here because it is useful in examples.

## DEFINED FUNCTIONS

Introduction. It would be impracticable and confusing to attempt to include as primitives in a language all of the functions which might prove useful in diverse areas of application. On the other hand, in any particular application there are many functions of general utility whose use should be made as convenient as possible. This need is met by the ability to define and name new functions, which can then be used with the convenience of primitives.

This section introduces the basic notions of function definition and illustrates the use of defined functions. Most of the detailed mechanics of function definition, revision, and display, are deferred to the succeeding section.

The sequence

```
      ∇SPHERE
[1]   SURF←4×3.14159×R×R
[2]   VOL←SURF×R÷3
[3]   ∇
```

is called a function definition; the first ∇ (pronounced del) marks the beginning of the definition and the second ∇ marks the conclusion: the name following the first ∇ (in this case SPHERE) is the name of the function defined, the numbers in brackets are statement numbers, and the accompanying statements form the body of the function definition.

The act of defining a function neither executes nor checks for validity the statements in the body; what it does is make the function name thereafter equivalent to the body. For example:

```
      ∇SPHERE                 Definition of the
[1]   SURF←4×3.14159×R×R      function SPHERE
[2]   VOL←SURF×R÷3
[3]   ∇
      R←2                     Specification and display
      R                       of the argument R

      SURF                    SURF has not been
VALUE ERROR                   assigned a value
      SURF
      ∧
      SPHERE                  Execution of SPHERE
      SURF                    SURF and VOL now have
50.26544                      values assigned by the
      VOL                     execution of SPHERE
33.51029333
      R←1                     Use of SPHERE for
      SPHERE                  a new value of the
      SURF                    argument R
12.56636
      VOL
4.188786667
```

Branching. Statements in a function are normally executed in the order indicated by the statement numbers, and execution terminates at the end of the last statement in the sequence. This normal order can be modified by branches. Branches make possible the construction of iterative procedures.

The expression →4 denotes a branch to statement 4 and causes statement 4 of the function to be executed next. In general, the arrow may be followed by any expression which, to be effective, must evaluate to an integer. This value is the number of the statement to be executed next. If the integer lies outside the range of statement numbers of the body of the function, the branch ends the execution of the function.

If the value of the expression to the right of a branch arrow is a non-empty vector, the branch is determined by its first component. If the vector is empty (i.e., of zero dimension) the branch is vacuous and the normal sequence is followed.

The following examples illustrate various methods of branching used in three equivalent functions (*SUM*, *SUM1*, and *SUM2*) for determining *S* as the sum of the first *N* integers:

```
      ∇SUM
[1]   S←0
[2]   I←1
[3]   →4×⍳I≤N          Branch to 4×1 (i.e., 4) or to 4×0 (out)
[4]   S←S+I
[5]   I←I+1
[6]   →3               Unconditional branch to 3
[7]   ∇
      N←1
      SUM
      S
1
      N←2
      SUM
      S
3
      N←5
      SUM
      S
15
      ∇SUM1            Equivalent to SUM
[1]   S←0
[2]   I←1
[3]   →0×⍳I>N          Branch to 0(out) or continue to next
[4]   S←S+I              line since 0×⍳0 is an empty vector
[5]   I←I+1
[6]   →3               Unconditional branch to 3
[7]   ∇
      N←5
      SUM1
      S
15
      ∇SUM2            Equivalent to SUM
[1]   S←0
[2]   I←0
[3]   S←S+I
[4]   I←I+1
[5]   →3×⍳I≤N          Branch to 3 or fall through(and out)
[6]   ∇
```

From the last two functions in the foregoing example, it should be clear that the expression ×⍳ occurring in a branch may often be read as "if". For example, →3×⍳*I*≤*N* may be read as "Branch to 3 if *I* is less than or equal to *N*."



Local and global variables. A variable is normally global in the sense that its name has the same significance no matter what function or functions it may be used in. However, the iteration counter *I* occurring in the foregoing function *SUM* is of interest only during execution of the function; it is frequently convenient to make such a variable local to a function in the sense that it has meaning only during the execution of the function and bears no relation to any object referred to by the same name at other times. Any number of variables can be made local to a function by appending each (preceded by a semicolon) to the function header. Compare the following behavior of the function *SUM3*, which has a local variable *I*, with the behavior of the function *SUM2* in which *I* is global:

```
      ∇SUM3;I                ∇SUM2
[1]   S←0                [1]   S←0
[2]   I←0                [2]   I←0
[3]   S←S+I              [3]   S←S+I
[4]   I←I+1              [4]   I←I+1
[5]   →3×⍳I≤N            [5]   →3×⍳I≤N
[6]   ∇                  [6]   ∇
      I←20                     I←20
      N←5                      N←5
      SUM3                     SUM2
      S                        S
15                       15
      I                        I
20                       6
```

Since *I* is local to the function *SUM3*, execution of *SUM3* has no effect on the variable *I* referred to before and after the use of *SUM3*.

However, if the variable *K* is local to a function *F* then any function *G* used within *F* may refer to the same variable *K*, unless the name *K* is further localized by being made local to *G*. For further treatment of this matter, see the section on Homonyms.

**Explicit argument.** A function definition of the form

```
      ∇SPH X
[1]   SUR←4×3.14159×X×X
[2]   ∇
```

defines *SPH* as a function with an explicit argument; whenever such a function is used it must be provided with an argument. For example:

```
      SPH 2
      SUR
50.26544
      SPH 1
      SUR
12.56636
```

Any explicit argument of a function is automatically made local to the function; if *E* is any expression, then the effect of *SPH E* is to assign to the local variable *X* the value of the expression *E* and then execute the body of the function *SPH*. Except for having a value assigned initially, the argument variable is treated as any other local variable and, in particular, may be respecified within the function.

**Explicit result.** Each of the primitive functions produces a result and may therefore appear within compound expressions. For example, the expression ⍕Z produces an explicit result and may therefore appear in a compound expression such as *X*+⍕Z. A function definition of the form

```
      ∇Z←SP X
[1]   Z←4×3.14159×X×X
      ∇
```

defines *SP* as a function with an explicit result; the variable *Z* is local, and the value it assumes at the completion of execution of the body of the function is the explicit result of the function. For example:

```
      Q←3×SP 1
      Q
[illegible]
      [illegible]
      (SP [illegible])×R:3
[illegible]
```

**The forms of defined functions.** Functions may be defined with 2,1, or 0 explicit arguments and either with or without an explicit result. The form of header used to define each of these six types is shown in Table 3.3. Each of the six forms permits the appending of semicolons and names to introduce local variables. The names appearing in any one header must all be distinct; e.g., the header *Z*←*F Z* is invalid.

| Number of Arguments | Number of Results 0 | 1 |
|---|---|---|
| 0 | ∇*F* | ∇*Z*←*F* |
| 1 | ∇*F Y* | ∇*Z*←*F Y* |
| 2 | ∇*X F Y* | ∇*Z*←*X F Y* |

Table 3.3: FORMS OF DEFINED FUNCTIONS

It is not obligatory either for the arguments of a defined function to be used within the body, or for the result variable to be specified. A function definition which does not assign a value to the result variable will engender a **value error** report when it is used within a compound expression. This behavior permits a function to be defined with a restricted domain, by testing the argument(s) and branching out in certain cases without specifying a result. For example:

```
      ∇Z←SQRT X
[1]   →0×⍳X<0
[2]   Z←X*.57
      Q←SQRT 16
      Q
→
      Q←SQRT ¯16
VALUE ERROR
      Q←SQRT ¯16
        ∧
```

Use of defined functions. A defined function may be used in the same ways that a primitive function may. In particular, it may be used within the definition of another function. For example, the function *HYP* determines the hypotenuse of a right triangle of sides *A* and *B* by using the square root function *SQRT*:

```
      ∇Z←SQRT X
[1]   Z←X*.5
      ∇
      ∇H←A HYP B
[1]   H←SQRT (A*2)+B*2
      ∇
      5 HYP 12
13
```

A defined function must be used with the same number of arguments as appear in its header.

Recursive function definition. A function *F* may be used in the body of its own definition, in which case the function is said to be recursively defined. The following program *FAC* shows a recursive definition of the factorial function. The heart of the definition is statement 2, which determines factorial *N* as the product of *N* and *FAC N*-1, except for the case *N*=0 when it is determined (by statement 4) as 1:

```
      ∇Z←FAC N
[1]   →4×⍳N=0
[2]   Z←N×FAC N-1
[3]   →0
[4]   Z←1
      ∇
```

Trace control. A trace is an automatic type-out of information generated by the execution of a function as it progresses. In a complete trace of a function *P*, the number of each statement executed is typed out in brackets, preceded by the function name *P* and followed by the final value produced by the statement. The trace is useful in analyzing the behavior of a defined function, particularly during its design.

The tracing of *P* is controlled by the trace vector for *P*, denoted by *T*Δ*P*. If one types *T*Δ*P*←2 3 5 then statements 2,3,and 5 will be traced in any subsequent execution of *P*. More generally, the value assigned to the trace vector may be any vector of integers. Typing *T*Δ*P*←0 will discontinue tracing of *P*. A complete trace of *P* is set up by entering *T*Δ*P*←⍳*N*, where *N* is the number of statements in *P*.

MECHANICS OF FUNCTION DEFINITION

When a function definition is opened (by typing a ∇ followed by a header), the system automatically types successive statement numbers enclosed in brackets and accepts successive entries as the statements forming the body of the definition. The system is therefore said to be in definition mode, as opposed to the execution mode which prevails outside of function definition.

There are several devices which may be used during function definition to revise and display the function being defined. After function definition has been closed, there are convenient ways to re-open the definition so that these same devices may be used for further revision or display.

Labels. If a statement occurring in the body of a function definition is prefaced by a name and a colon, then at the end of the definition the name is assigned a value equal to the statement number. A variable specified in this way is called a label. Labels are used to advantage in branches when it is expected that a function definition may be changed for one reason or another, since a label automatically assumes the new value of the statement number of its associated statement as statements are inserted or deleted.

Revision. Any statement number (including one typed by the system) can be overridden by typing [N], where N is any positive number less than 10000, with or without a decimal point and with at most four digits to the right of the decimal point. If N is zero, it refers to the header line of the function.

If any statement number is repeated, the statement following it supersedes the earlier specification of the statement. If any statement is empty -- that is, the bracketed statement number was immediately followed by both a linefeed and a carriage return (a carriage return alone is vacuous) -- it is deleted.

When the function definition mode is ended, the statements are reordered according to their statement numbers and the statement numbers are replaced by the integers 1,2,3, and so on. Labels are assigned appropriate values.

The particular statement on which the closing ∇ appears is not significant, since it marks only the end of the definition mode, not necessarily the last line of the function. Moreover, the closing ∇ may be entered either alone or at the end of a statement.

Display. During function definition, statement N can be displayed by overriding the line number with [N□] After the display, the system awaits replacement of statement N. Typing [□] displays the entire function, including the header and the opening and closing ∇, and awaits entry of the next statement; typing [□N] displays all statements from N onward and awaits replacement of the last statement. Executing an attention signal will stop any display.

Line editing. During function definition, statement N can be modified by the following mechanism:

1. Type [N□M] where M is an integer.

2. Statement N is automatically displayed and the carriage stops under position M.

3. A letter or decimal digit or the symbol / may be typed under any of the positions in the displayed statement. Any other characters typed in this mode are ignored. The ordinary rules for backspace and linefeed apply.

4. When the carriage is returned, statement N is re-displayed. Each character understruck by a is deleted, each character understruck by a digit K is preceded by K added spaces, and each character understruck by a letter is preceded by $5+R$ spaces, where $R$ is the rank of the letter in the alphabet. Finally, the carriage moves to the first injected space and awaits the typing of modifications to the statement in the usual manner. The final effect is to define the statement exactly as if the entry had been made entirely from the keyboard; in particular, a completely blank sequence leaves the statement unchanged.

If the statement number itself is changed during the editing procedure, the statement affected is determined by the new statement number; hence statement N remains unchanged. This permits statements to be moved, with or without modification.

Reopening function definition. If a function $R$ is already defined, the definition mode for that function can be re-established by entering ∇R alone; the rest of the function header must not be entered. The system responds by typing [N+1], where $N$ is the number of statements in $R$. Function definition then proceeds in the normal manner.

Function definition may also be established with editing or display requested on the same line. For example, ∇R[3]X←X+1 initiates editing by entering a new line 3 immediately. The system responds by typing [4] and awaiting continuation. The entire process may be accomplished on a single line. Thus, ∇R[3]X←X+1∇ opens the definition of $R$, enters a new line 3, and terminates the definition mode. Also, ∇R[□]∇ causes the entire definition of $R$ to be displayed, after which the system returns to execution mode.

Similar expressions involving display are also permissible, for example, ∇R[□3]∇ or ∇R[□] or ∇R[.□10].

Locked functions. If the symbol ⍫ (formed by a ∇ overstruck with a ~ and called del-tilde) is used instead of ∇ to open or close a function definition, the function becomes locked. A locked function cannot be revised or displayed in any way. Moreover, an error stop within the function will print only the function name and statement number, not the statement. Finally, the associated stop control (see next section) and trace control vectors cannot be changed after the function is locked.

Locked functions are used to keep a function definition proprietary. For example, in an exercise in which a student is required to determine the behavior of a function by using it for a variety of arguments, locking a function prevents him from displaying its definition.

Deletion of functions and variables. A function (whether locked or not) is deleted by the command )ERASE (see Table 2.1). It may also be deleted by deleting every one of its statements. A variable may be deleted only by the erase command.

System command entered during function definition. A system command entered during function definition will not be accepted as a statement in the definition. Some commands, such as *)COPY*, will be rejected with the message *NOT WITH OPEN DEFINITION* (see Table 2.1); most will be executed immediately.

SUSPENDED FUNCTION EXECUTION

Suspension. The execution of a function *F* may be stopped before completion in a variety of ways: by an error report, by an attention signal, or by the stop control vector *SΔF* treated below. In any case, the function is still active and its execution can later be resumed. In this state the function is said to be suspended. Typing →K will cause execution of the suspended function to be resumed, beginning with statement K.

Whatever the reason for suspension, the statement or statement number displayed is the next one to have been executed. A branch to that statement number will cause normal continuation of the function execution, and a branch out (→0) will terminate execution of the function.

The function ⌶26 (described in the section on System Dependent Functions) yields the number of the statement next to be executed. Hence the expression →⌶26 provides a safe and convenient way to cause normal resumption of execution.

In the suspended state all normal activities are possible. In particular, the system is in a condition to:

1. execute statements or system commands.

2. resume execution of the function at an arbitrary point *N* (by entering →*N*).

3. reopen the definition of any function which is not pendent. The term pendent is defined in the discussion of the state indicator below.

If function execution is interrupted by a disconnect, the function is suspended and the resulting active workspace is automatically saved under the name *CONTINUE*, as noted in Part 2.

State indicator. Typing *)SI* causes a display of the state indicator; a typical display has the following form:

```
      )SI
H[7]  *
G[2]
F[3]
```

The foregoing display indicates that execution was halted just before executing statement 7 of the function *H*, that the current use of function *H* was invoked in statement 2 of function *G*, and that the use of function *G* was in turn invoked in statement 3 of *F*. The * appearing to the right of *H*[7] indicates that the function *H* is suspended; the functions *G* and *F* are said to be pendent.

Further functions can be invoked when in the suspended state. Thus if *G* were now invoked and a further suspension occurred in statement 5 of *H*, itself invoked in statement 8 of *G*, a subsequent display of the state indicator would appear as follows:

```
      )SI
H[5]  *
G[8]
H[7]  *
G[2]
F[3]
```

The entire sequence from the last to the preceding suspension can be cleared by typing a branch with no argument (that is, →). This behavior is illustrated by continuing the foregoing example as follows:

```
      →
      )SI
H[7]  *
G[2]
F[3]
```

Repeated use of → will clear the state indicator completely. The cleared state indicator displays as a blank line.

Stop Control. The stop vector for a function *P* is denoted by *SΔP*. It is set in the same manner as the trace vector (i.e., by *SΔP←I*, where the vector *I* specifies the numbers of the statements controlled), and stops execution just before each of the specified statements. At each stop, the function name and the line number of the statement next to be executed are printed. After the stop the system is in the normal suspended state; resumption of execution may therefore be initiated by a branch.

Trace control and stop control can be used in conjunction. Moreover, either of the control vectors may be set within functions. In particular, they may be set by expressions which initiate tracing or halts only for certain values of certain variables.

## HOMONYMS

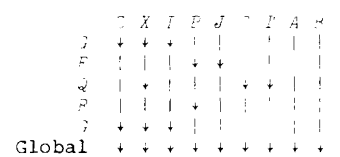
Variable names. The use of local variables introduces the possibility of having more than one object in a workspace with the same name. Confusion is avoided by the following rule: when a function is executed, its local variables supersede, for the duration of the execution, other objects of the same name. A name may, therefore, be said to have one active referent and (possibly) several latent referents.

The complete set of referents of a name can be determined with the aid of the SIV list (state indicator with local variables), whose display is initiated by the command )*SIV*. The SIV list contains the information provided by the command )*SI*, augmented by the names of the variables local to each function. A sample display follows:

```
      )SIV
Z[7]  * Z X I
F[4]    P J
G[3]  * G X T
H[2]    P
G[3]    G X I
```

If the SIV list is scanned downward, from the top, the first occurrence of a variable is the point at which its active referent was introduced; lower occurrences are the points at which currently latent referents were introduced; and if the name is not found at all, its referent is global, and should be sought for with the commands )*FNS*, )*VARS*, or )*CRIC*.

As the state indicator is cleared (by →, or by the continuation to completion of halted functions), latent referents become active in the sequence summarized, for the preceding SIV list, by the following diagram:




The currently active referent of a name holds down to and including the execution of the function listed at the point of the first arrow, because of localization of the name within that function. The first latent referent becomes active when that function is completed, and holds down to the next arrow; and so forth until the state indicator is completely cleared, at which point there are no longer any latent referents, and all active referents are global objects.

Function names. All function names are global. In the foregoing example, therefore, a function named *P* cannot be used within the function *R* or within any of the functions employed by *R*, since the local variable name *P* makes the function *P* inaccessible. However, even in such circumstances, the opening of function definition for such a function *P* is possible. (Moreover, as stated in Part 2, system commands concern global objects only, regardless of the current environment.)

This scheme of homonyms is easy to use and relatively free from pitfalls. It can, however, lead to seeming anomalies as indicated by the following example (shown to the authors by J.C.Shaw) of two pairs of functions which differ only in the name used for the argument:

```
      ∇Z←F X
[1]   Z←X+Y∇

      ∇Z←G Y
[1]   Z←F Y∇
      Y←3
      G 4
3

      ∇J←F X
[1]     ←X+Y∇

      ∇Z←G A
[1]   Z←F A∇
      Y←3
      G 4
7
```

## INPUT AND OUTPUT

The following function determines the value of an amount *A* invested at interest *B*[1] for a period of *B*[2] years:

```
      ∇Z←A CPI B
[1]   Z←A×(1+.01×B[1])*B[2]∇
```

For example:

```
      1000 CPI 5 4
1215.506250
```

The casual user of such a function might, however, find it onerous to remember the positions of the various arguments and whether the interest rate is to be entered as the actual rate (e.g., .05) or in percent (e.g., 5). An exchange of the following form might be more palatable:

```
      CI
ENTER CAPITAL AMOUNT IN DOLLARS
⎕:
      1000
ENTER INTEREST IN PERCENT
⎕:
      5
ENTER PERIOD IN YEARS
⎕:
      4
RESULT IS 1215.50625
```

It is necessary that each of the keyboard entries (1000, 5, and 4) occurring in such an exchange be accepted not as an ordinary entry (which would only evoke the response 1000, etc.), but as data to be used within the function *CI*. Facilities for this are provided in two ways, termed evaluated input, and character input.

The definition of the function *CI* is shown at the end of this section.

Evaluated input. The quad symbol ⎕ appearing anywhere other than immediately to the left of a specification arrow accepts keyboard input as follows: the two symbols ⎕: are printed, the paper is spaced up one line, and the keyboard unlocks. Any valid expression entered at this point is evaluated and the result is substituted for the quad. For example:

```
      ∇Z←F
[1]   Z←4×⎕*2
[2]   ∇
      F
⎕:
      3
36
      F
⎕:
      3÷2
9
```

An invalid entry in response to request for a quad input induces an appropriate error report, after which input is again awaited at the same point. A system command entered will be executed, after which (except in the case of one which replaces the active workspace) a valid expression will again be awaited. An empty input (i.e., a carriage return alone or spaces and a carriage return) is rejected and the system again prints the symbols ⎕: and awaits input.

The symbols ⎕: are printed to alert the user to the type of input expected; they can be changed by the library function *SETI* as described in Part 4.

Character input. The quote-quad symbol ⍞ (i.e., a quad overstruck with a quote) accepts character input: the keyboard unlocks at the left margin and data entered are accepted as characters. For example:

```
      X←⍞
CAN'T                (Quote-quad input, not indented)
      X
CAN'T
```

Escape from input loop. If evaluated or character input occurs within an endless loop in a function, it may be impossible to escape by the usual device of striking the attention button. Escape from ⎕ input can be achieved by entering →. Escape from ⍞ input can be achieved by typing the three letters *OUT*, in that order, but with a backspace between each pair so that they all overstrike. The effect is exactly as if the symbol → were entered while suspended.

Normal output. The quad symbol appearing immediately to the left of a specification arrow indicates that the value of the expression to the right of the arrow is to be printed. Hence, ⎕←*X* is equivalent to the statement *X*. The longer form ⎕← *X* is useful when employing multiple specification. For example, ⎕←*Q*←*X**2 assigns to *Q* the value *X**2 and then prints the value of *X**2.

The page width (measured in characters) may be set to any value *N* in the range 30-130 by entering the command *)WIDTH N*. It may also be set by the library function *WIDTH* which may be used within a defined function. (See Part 4.)

Heterogeneous output. A sequence of expressions separated by semi-colons will cause the values of the expressions to be printed, with no intervening carriage returns or spaces except those implicit in the display of the values.

The primary use of this form is for output in which some of the expressions yield numbers and some yield characters. For example, if X←2 14 , then:

```
      'THE VALUE OF X IS ';X
THE VALUE OF X IS  2 14
```

A further example of mixed output is furnished by the definition of the function CI which introduced the present section:

```
      ∇CI;A;I;Y
[1]   'ENTER CAPITAL AMOUNT IN DOLLARS'
[2]   A←□
[3]   'ENTER INTEREST IN PERCENT'
[4]   I←□
[5]   'ENTER PERIOD IN YEARS'
[6]   Y←□
[7]   'RESULT IS ';A×(1+.01×I)*Y∇
```

RECTANGULAR ARRAYS

Introduction. A single element of a rectangular array can be selected by specifying its indices; the number of indices required is called the dimensionality or rank of the array. Thus a vector is of rank 1, a matrix (in which the first index selects a row and the second a column) is of rank 2, and a scalar (since it permits no selection by indices) is an array of rank 0. Rectangular arrays of higher rank may be used, and are called 3-dimensional, 4-dimensional, etc.

This section treats the reshaping and indexing of arrays, and the form of array output. The following section treats the four ways in which the basic scalar functions are extended to arrays, and the next section thereafter treats the definition of certain mixed functions on arrays.

Vectors, dimension, catenation. If X is a vector, then ⍴X denotes its dimension. For example, if X←2 3 5 7 11, then ⍴X is 5, and if Y←'ABC' , then ⍴Y is 3. A single character entered in quotes or in response to a □ input is a scalar, not a vector of dimension 1; this parallels the case of a single number, which is also a scalar.

Catenation chains two vectors (or scalars) together to form a vector; it is denoted by a comma. For example:

```
      X←2 3 5 7 11
      X,X
2 3 5 7 11 2 3 5 7 11
```

In general, the dimension of X,Y is equal to the total number of elements in X and Y. A numeric vector cannot be catenated with a character vector. (However, see Heterogeneous Output.)

Matrices, dimension, ravel. The monadic function ⍴ applied to an array A yields the size of A, that is, a vector whose components are the dimensions of A. For example, if M is the matrix

```
 1  2  3  4
 5  6  7  8
 9 10 11 12
```

of three rows and four columns, then ⍴M is the vector 3 4.

Since ⍴A contains one component for each coordinate of A, the expression ⍴⍴A is the rank of A. Table 3.4 illustrates the values of ⍴A and ⍴⍴A for arrays of rank 0 (scalars) up to rank 3. In particular, the function applied to a scalar yields an empty vector.

| Type of Array | ⍴A | ⍴⍴A | ⍴⍴⍴A |
|---|---|---|---|
| Scalar | | 0 | 1 |
| Vector | V | 1 | 1 |
| Matrix | M | 2 | 1 |
| 3-Dimensional | T | 3 | 1 |

Table 3.4: DIMENSION AND RANK VECTORS

The monadic function ravel is denoted by a comma; when applied to any array $A$ it produces a vector whose elements are the elements of $A$ in row order. For example, if $A$ is the matrix

```
 2  4  6  8
10 12 14 16
18 20 22 24
```

and if $V\leftarrow,A$ then $V$ is a vector of dimension 12 whose elements are the integers `2 4 6 8 10 12 ... 24`. If $A$ is a vector, then $,A$ is equivalent to $A$; if $A$ is a scalar, then $,A$ is a vector of dimension 1.

Reshape. The dyadic function $\rho$ reshapes its right argument to the dimension specified by its left argument. If $M\leftarrow D\rho V$, then $M$ is an array of dimension $D$ whose elements are the elements of $V$. For example, `2 3⍴1 2 3 4 5 6` is the matrix

```
1 2 3
4 5 6
```

If $N$, the total number of elements required in the array $D\rho V$, is equal to the dimension of the vector $V$, then the ravel of $D\rho V$ is equal to $V$. If $N$ is less than $\rho V$, then only the first $N$ elements of $V$ are used; if $N$ is greater than $\rho V$, then the elements of $V$ are repeated cyclically. For example, `2 3⍴1 2` is the matrix

```
1 2 1
2 1 2
```

and `3 3⍴1 0 0 0` is the identity matrix

```
1 0 0
0 1 0
0 0 1
```

More generally, if $A$ is any array, then $,\rho A$ is equivalent to $B\rho,A$. For example, if $A$ is the matrix

```
1 2 3
4 5 6
```

then `3 5⍴A` is the matrix

```
1 2 3 4 5
6 1 2 3 4
5 6 1 2 3
```

The expressions $0\rho X$ and `0 3⍴X` and `3 0⍴X` and `0 0⍴X` are all valid; any one or more of the dimensions of an array may be zero.

Uses of empty arrays. A vector of dimension zero contains no components and is called an empty vector. Three expressions which yield empty vectors are `⍳0` and `''` and $\rho$ applied to any scalar. An empty vector prints as a blank line.

One important use of the empty vector has already been illustrated: when one occurs as the argument of a branch, the effect is to continue the normal sequence.

The following function for determining the representation of any positive integer $N$ in a base $B$ number system shows a typical use of the empty vector in initializing a vector $Z$ which is to be built up by successive catenations:

```
      ∇Z←B BASE N
[1]   Z←⍳0
[2]   Z←(B|N),Z
[3]   N←⌊N÷B
[4]   →2×N>0⍴⍴N
      10 BASE 1776
1 7 7 6
      8 BASE 1776
3 3 6 0
```

Empty arrays of higher rank can be useful in analogous ways in conjunction with the expansion function described in the section on Mixed Functions.

**Indexing.** If $X$ is a vector and $I$ is a scalar, then $X[I]$ denotes the $I$th element of $X$. For example, if $X \leftarrow 2\ 3\ 5\ 7\ 11$ then $X[2]$ is 3.

If the index $I$ is a vector, then $X[I]$ is the vector obtained by selecting from $X$ the elements indicated by successive components of $I$. For example, $X[1\ 3\ 5]$ is $2\ 5\ 11$ and $X[5\ 4\ 3\ 2\ 1]$ is $11\ 7\ 5\ 3\ 2$ and $X[3]$ is $2\ 3\ 5$. If the elements of $I$ do not belong to the set of indices of $X$, then the expression $X[I]$ induces an **index error** report.

In general, $\rho X[I]$ is equal to $\rho I$. In particular, if $I$ is a scalar, then $X[I]$ is a scalar, and if $I$ is a matrix, then $X[I]$ is a matrix. For example:

```
      A←'ABCDEFG'
      M←4 3⍴3 1 4 2 1 4 4 1 2 4 1 4
      M
3 1 4
2 1 4
4 1 2
4 1 4
      A[M]

CAD
BAD
CAB
BAD
```

If $M$ is a matrix, then $M$ is indexed by a two-part list of the form $I;J$ where $I$ selects the row (or rows) and $J$ selects the column (or columns). For example, if $M$ is the matrix

```
 1  2  3  4
 5  6  7  8
 9 10 11 12
```

then $M[2;3]$ is the element 7 and $M[1\ 3;\ 2\ 3\ 4]$ is the matrix

```
 2  3  4
10 11 12
```

In general, $\rho M[I;J]$ is equal to $(\rho I),\rho J$. Hence if $I$ and $J$ are both vectors, then $M[I;J]$ is a matrix; if both $I$ and $J$ are scalars, $M[I;J]$ is a scalar; if $I$ is a vector and $J$ is a scalar (or vice versa), $M[I;J]$ is a vector. The indices are not limited to vectors, but may be of higher rank. For example, if $I$ is a 3 by 4 matrix, and $J$ is a vector of dimension 6, then $M[I;J]$ is of dimension $3\ 4\ 6$, and $M[J;I]$ is of dimension $6\ 3\ 4$. In particular if $P$ and $Q$ are matrices, and if $R \leftarrow T[P;Q]$, then $R$ is an array of rank 4 and $R[I;J;K;L]$ is equal to $T[P[I;J];Q[K;L]]$.

The form $M[I;]$ indicates that all columns are selected, and the form $M[;J]$ indicates that all rows are selected. For example, $M[2;]$ is $5\ 6\ 7\ 8$ and $M[;2\ 1]$ is

```
 2  1
 6  5
10  9
```

The following example illustrates the use of a matrix indexing a matrix to obtain a three-dimensional array:

```
      M←4 3⍴3 1 4 2 1 4 4 1 2 4 1 4
      M
3 1 4
2 1 4
4 1 2
4 1 4
      M[M;]

4 1 2
3 1 4
4 1 4

2 1 4
3 1 4
4 1 4

4 1 4
3 1 4
2 1 4

4 1 4
3 1 4
4 1 4
```

Permutations are an interesting use of indexing. A vector *P* whose elements are some permutation of its own indices is called a permutation of order ⍴*P*. For example, 3 1 4 2 is a permutation of order 4. If *X* is any vector of the same dimension as *P*, then *X*[*P*] produces a permutation of *X*. Moreover, if ⍴*P* is equal to (⍴*M*)[1], then *M*[*P*;] permutes the column vectors of *M* (i.e., interchanges the rows of *M*) and is called a column permutation. Similarly, if ⍴*P* equals (⍴*M*)[2], then *M*[;*P*] is a row permutation of *M*.

Indexing on the left. An array appearing to the left of a specification arrow may be indexed, in which case only the selected positions are affected by the specification. For example:

```
      X←2 3 5 7 11
      X[1 3]←6 8
      X
6 3 8 7 11
```

The normal restrictions on indexing apply; in particular, a variable which has not already been assigned a value cannot be indexed, and an out-of-range index value cannot be used.

Index origin. In 1-origin indexing, *X*[1] is the leading element of the vector *X* and *X*[⍴*X*] is the last element. In 0-origin indexing, *X*[0] is the leading element and *X*[¯1+⍴*X*] is the last. 0-origin indexing is instituted by the command *)ORIGIN 0*. The command *)ORIGIN 1* restores 1-origin indexing. The index origin in effect applies to all coordinates of all rectangular arrays.

The function *ORIGIN* in Library 1 *WSFNS* may also be used to control the index origin. It may be executed within a function. (See Part 4.)

In certain expressions such as +/[*J*]*M* and *K*⌽[*J*]*M* (to be treated more fully in the two following sections), the value of *J* determines the coordinate of the array *M* along which the function is to be applied. Since the numbering of coordinates follows the index origin, a change of index origin also affects the behavior of such expressions.

The index origin also affects four other functions, the monadic and the dyadic forms of ? and ⍳. The expression ⍳*N* yields a vector of the first *N* integers beginning with the index origin. Hence *X*[⍳*N*] selects the first *N* components of *X* in either origin. Moreover, ⍳1 is a one-element vector having the value 0 in 0-origin and 1 in 1-origin; ⍳0 is an empty vector in either origin.

The index origin remains associated with a workspace; in particular, the index origin of an active workspace is not affected by a copy command. A clean workspace provided on sign-on or by the command )CLEAR is in 1-origin. All definitions and examples in this text are expressed in 1-origin.

Array output. Character arrays print with no spaces between components in each row; other arrays print with at least one space. If a vector or a row of a matrix requires more than one line, succeeding lines are indented.

A matrix prints with all columns aligned and with a blank line before the first row. A matrix of dimension *N*,1 prints as a single column.

## FUNCTIONS ON ARRAYS

There are four ways in which the scalar functions of Table 3.2 extend to arrays: element-by-element, reduction, inner product, and outer product. Reduction and outer product are defined on any arrays, but the other two extensions are defined only on arrays whose sizes satisfy a certain relationship called conformability. For the element-by-element extension, conformability requires that the shapes of the arrays agree, unless one is a scalar. The requirements for inner product are shown in Table 3.6.

Scalar functions. All of the scalar functions of Table 3.2 are extended to arrays element by element. Thus if *M* and *N* are matrices of the same size, f is a scalar function, and *P*←*M*f*N*, then *P*[*I*;*J*] equals *M*[*I*;*J*]f*N*[*I*;*J*], and if *Q*←f*N*, then *Q*[*I*;*J*] is equal to f*N*[*I*;*J*]

If *M* and *N* are not of the same size, then *M*f*N* is undefined (and induces a length or rank error report) unless one or other of *M* and *N* is a scalar or one-element array, in which case the single element is applied to each element of the other argument. In particular, a scalar versus an empty array produces an empty array.

An expression or function definition which employs only scalar functions and scalar constants extends to arrays like a scalar function.

Reduction. The sum-reduction of a vector *X* is denoted by +/*X* and defined as the sum of all components of *X*. More generally, for any scalar dyadic function f, the expression f/*X* is equivalent to *X*[1]f*X*[2]f. f*X*[⍴*X*], where evaluation is from rightmost to leftmost as usual. A user-defined function cannot be used in reduction.

If $X$ is a vector of dimension zero, then f/$X$ yields the identity element of the function f (listed in Table 3.5) if it exists; if $X$ is a scalar or a vector of dimension 1, then f/$X$ yields the value of the single element of $X$.

The result of reducing any vector or scalar is a scalar.

| Dyadic Function | | Identity Element | Left-Right |
|---|---|---|---|
| Times | × | 1 | L R |
| Plus | + | 0 | L R |
| Divide | ÷ | 1 | R |
| Minus | - | 0 | R |
| Power | * | 1 | R |
| Logarithm | ⍟ | | None |
| Maximum | ⌈ | ¯7.237 . E75 | L R |
| Minimum | ⌊ | 7.237 E75 | L R |
| Residue | \| | 0 | L |
| Circle | ○ | | None |
| Out of | ! | 1 | L |
| Or | ∨ | 0 | L R |
| And | ∧ | 1 | L R |
| Nor | ⍱ | | None |
| Nand | ⍲ | | None |
| Equal | = | 1 } Apply | L R |
| Not equal | ≠ | 0 } for | L R |
| Greater | > | 0 } logical | R |
| Not less | ≥ | 1 } arguments | R |
| Less | < | 0 } only | L |
| Not greater | ≤ | 1 } | L |

Table 3.5: IDENTITY ELEMENTS OF PRIMITIVE SCALAR DYADIC FUNCTIONS

For a matrix $M$, reduction can proceed along the first coordinate (denoted by f⌿$M$) or along the second coordinate (f⌿[2]$M$). The result in either case is a vector; in general, reduction applied to any non-scalar array $A$ produces a result of rank one less than the rank of $A$ (hence the term reduction). The numbering of coordinates follows the index origin, and an attempt to reduce along a non-existent coordinate will result in an index error.

Since +/[1]$M$ scans over the row index of $M$ it sums each column vector of $M$, and +/[2]$M$ sums the row vectors of $M$. For example, if $M$ is the matrix

```
1  2  3
4  5  6
```

then +/[1]$M$ is `5 7 9` and +/[2]$M$ is `6 15`.

In reducing along the last coordinate of an array, the coordinate indicator may be elided -- thus +/$M$ denotes summing over each of the rows of $M$ and +/$V$ denotes summing over the last (and only) coordinate of the vector $V$.

Reduction over the first coordinate of $M$ by a function f may be obtained by using the expression f⌿$M$. The symbol ⌿ is formed by overstriking the solidus with the minus sign.

Inner product. The familiar matrix product is denoted by $C$←$A$+.×$B$. If $A$ and $B$ are matrices, then $C$ is a matrix such that $C[I;J]$ is equal to +/$A[I;]$×$B[;J]$. A similar definition applies to $A$f.g$B$ where f and g are any of the standard scalar dyadic functions.

If $A$ is a vector and $B$ is a matrix, then $C$ is a vector such that $C[J]$ is equal to +/$A$×$B[;J]$. If $B$ is a vector and $A$ is a matrix, then $C$ is a vector such that $C[I]$ is equal to +/$A[I;]$×$B$. If both $A$ and $B$ are vectors, then $A$+.×$B$ is the scalar +/$A$×$B$.

The last dimension of the pre-multiplier $A$ must equal the first dimension of the post-multiplier $B$, except that if either argument is a scalar, it is extended in the usual way. For non-scalar arguments, the dimension of the result is equal to (¯1↓⍴$A$),1↓⍴$B$. (See the function drop in the section on Mixed Functions.) In other words, the dimension of the result is equal to (⍴$A$),⍴$B$ except for the two inner dimensions (¯1↑⍴$A$ and 1↑⍴$B$), which must agree and which are eliminated by the reduction over them.

Definitions for various cases are shown in Table 3.6.

Outer product. The outer product of two arrays $X$ and $Y$ with respect to a standard scalar dyadic function g is denoted by $X$∘.g$Y$ and yields an array of dimension (⍴$X$),⍴$Y$, formed by applying g to every pair of components of $X$ and $Y$.

| ρA | ρB | ρA f.g B | Conformability requirements | Definition Z←A f.g B |
|---|---|---|---|---|
| | | | | Z←f/A g B |
| | V | | | Z←f/A g B |
| U | | | | Z←f/A g B |
| U | V | | U≡V | Z←f/A g B |
| | V W | W | | Z[I]←f/A g B[;I] |
| T U | | T | | Z[I]←f/A[I;] g B |
| U | V W | W | U≡V | Z[I]←f/A g B[;I] |
| T U | V | T | U≡V | Z[I]←f/A[I;] g B |
| T U | V W | T W | U≡V | Z[I;J]←f/A[I;] g B[;J] |

Table 3.6: INNER PRODUCTS FOR PRIMITIVE SCALAR DYADIC FUNCTIONS f AND g

If $X$ and $Y$ are vectors and $Z\leftarrow X\circ.gY$, then $Z[I;J]$ is equal to $X[I]gY[J]$. For example:

```
      X←⍳3
      Y←⍳4
      X∘.×Y
 1  2  3  4
 2  4  6  8
 3  6  9 12
      X∘.≥Y
1 0 0 0
1 1 0 0
1 1 1 0
```

If $X$ is a vector and $Y$ is a matrix, and $Z\leftarrow X\circ.gY$, then $Z[I;J;K]$ is equal to $X[I]gY[J;K]$. Definitions for various cases are shown in Table 3.7.

| ρA | ρB | ρA∘.gB | Definition Z←A∘.gB |
|---|---|---|---|
| | | | Z←A g B |
| | V | V | Z[I]←A g B[I] |
| U | | U | Z[I]←A[I] g B |
| U | V | U V | Z[I;J]←A[I] g B[J] |
| | V W | V W | Z[I;J]←A g B[I;J] |
| T U | | T U | Z[I;J]←A[I;J] g B |
| U | V W | U V W | Z[I;J;K]←A[I] g B[J;K] |
| T U | V | T U V | Z[I;J;K]←A[I;J] g B[K] |
| T U | V W | T U V W | Z[I;J;K;L]←A[I;J] g B[K;L] |

Table 3.7: OUTER PRODUCTS FOR PRIMITIVE SCALAR DYADIC FUNCTION g

## MIXED FUNCTIONS

Introduction. The scalar functions listed in Table 3.2 each take a scalar argument (or arguments) and yield a scalar result; each is also extended element by element to arrays. The mixed functions of Table 3.8, on the other hand, may be defined on vector arguments to yield a scalar result or a vector result, or may be defined on scalar arguments to yield a vector result. In extending these definitions to arrays of higher rank, it may therefore be necessary to specify which coordinate of an array the mixed function is applied to. The expression $[J]$ following a function symbol indicates that the function is applied to the $J$th coordinate. If the expression is elided, the function applies to the last coordinate of the argument array. These conventions agree with those used earlier in reduction.

The numbering of coordinates follows the index origin.

Transpose. The expression 2 1⍉M yields the transpose of the matrix $M$; that is, if $R\leftarrow$2 1⍉$M$, then each element $R[I;J]$ is equal to $M[J;I]$. For example:

```
      M                  2 1⍉M
 1  2  3  4            1  5  9
 5  6  7  8            2  6 10
 9 10 11 12            3  7 11
                       4  8 12
```

If $P$ is any permutation of order ⍴⍴A, then P⍉A is an array similar to $A$ except that the coordinates are permuted: the $I$th coordinate becomes the $P[I]$th coordinate of the result. Hence, if $R\leftarrow P⍉A$, then $(\rho R)[P]$ is equal to $\rho A$. For example:

```
      A←2 3 5 7⍴⍳210
      ⍴A
2 3 5 7
      P←2 3 4 1
      ⍴P⍉A
7 2 3 5
```

| Name | Sign[1] | Definition or example[2] |
|---|---|---|
| Size | ρA | ρP ↔ 4  ρE ↔ 3 4  ρ5 ↔ ι0 |
| Reshape | VρA | Reshape A to dimension V  3 4ρι12 ↔ E<br>12ρE ↔ ι12  0ρE ↔ ι0 |
| Ravel | ,A | ,A ↔ (×/ρA)ρA  ,E ↔ι12  ρ,5 ↔ 1 |
| Catenate | V,V | P,ι2 ↔ 2 3 5 7 1 2  'T','HIS' ↔ 'THIS' |
| Index[3 4] | V[A]<br>M[A;A]<br>A[A;<br>. ;A] | P[2] ↔3  P[4 3 2 1] ↔7 5 3 2<br>E[1 3;3 2 1] ↔ 3 2 [illegible]<br>11 10 [illegible]<br>E[1;] ↔ 1 2 3 4  ABCD<br>E[;1] ↔ 1 5 9  'ABCDEFGHIJKL'[E] ↔ EFGH<br>IJKL |
| Index generator[3] | ιS | First S integers  ι4 ↔ 1 2 3 4<br>ι0 ↔ an empty vector |
| Index of[3] | VιA | Least index of A in V, or 1+ρV  Pι3 ↔2  5 1 2 5<br>PιE ↔ 3 5 4 [illegible]<br>4 4ι4 ↔ 1  5 5 5 5 |
| Take<br>Drop | V↑A<br>V↓A | Take (drop) ↑V[I] first elements on coordinate I. (Last if V[I]<0)  [illegible] |
| Grade up[5]<br>Grade down[5] | ⍋A<br>⍒A | The permutation which would order A (ascending or descending)  ⍋3 5 3 ↔ 4 1 3 2<br>⍒3 5 3 2 ↔ 2 1 3 4 |
| Compress[5] | V/A | 1 0 1 0/P ↔ 2 5  1 0 1 0/E ↔ 1 3 / 5 7 / 9 11<br>1 0 1/[1]E ↔ 1 2 3 4 ↔ 1 0 1≠E<br>9 10 11 12 |
| Expand[5] | V\A | 1 0 1\ι2 ↔ 1 0 2  1 0 1 1 1\X ↔ A [illegible] / E FGH / I JKL |
| Reverse[5] | ⌽A | DCBA<br>⌽X ↔ HGFE  ⌽[1]X ↔ ⊖X ↔ [illegible]<br>LKJI  ⌽E ↔ [illegible] |
| Rotate[5] | A⌽A | 3⌽E ↔ [illegible]  3 5 ↔ ¯1⌽E  1 ¯1⌽X ↔ [illegible] |
| Transpose | V⍉A<br>⍉A | Coordinate I of A becomes coordinate V[I] of result  1⍉X ↔ [illegible]<br>1 1⍉E ↔ [illegible]<br>Transpose last two coordinates  ⍉[illegible] ↔ [illegible] |
| Membership | A∊A | ρW∊Y ↔ ρW  [illegible] ↔ 0 1 1 0<br>P∊ι4 ↔ 1 1 0 [illegible]  [illegible] |
| Decode | V⊥V | 10⊥1 [illegible] 6 ↔ 17  [illegible] |
| Encode | V⊤S | 24 60 60⊤3723 ↔ 1 2 3  [illegible] ↔ 2 [illegible] |
| Deal[3] | [illegible] | W?Y ↔ Random deal of W elements from |

Table 3.8: PRIMITIVE MIXED FUNCTIONS (see adjacent notes)

1. Restrictions on argument ranks are indicated by: S for scalar, V for vector, M for matrix, A for Any. Except as the first argument of SιA or S[A], a scalar may be used instead of a vector. A one-element array may replace any scalar.

2. Arrays used in examples: P ↔ 2 3 5 7  E ↔ 1 2 3 4 / 5 6 7 8 / 9 10 11 12  X ↔ ABCD / EFGH / IJKL

3. Function depends on index origin.

4. Elision of any index selects all along that coordinate.

5. The function is applied along the last coordinate; the symbols ≠, ⍀, and ⊖ are equivalent to /, \, and ⌽, respectively, except that the function is applied along the first coordinate. If [S] appears after any of the symbols, the relevant coordinate is determined by the scalar S.

Notes to Table 3.8

More generally, Q⍉A is a valid expression if Q is any vector of dimension ρρA whose elements are chosen from (and exhaust) the elements of ιf/Q. For example, if ρρA is equal to 3, then 1 1 2 and 2 1 1 and 1 1 1 are suitable values for Q but 1 3 1 is not. Just as for the case P⍉A where P is a permutation vector, the Ith coordinate becomes the V[I]th coordinate of Q⍉A. However, in this case two or more of the coordinates of A may map into a single coordinate of the result, thus producing a diagonal section of A as illustrated below:

```
      A←3 3ρι9
      A
1     3
[illegible]
7     [illegible]
      1 1⍉A
[illegible]
```

Table 3.9 shows the detailed definitions of transposition for a variety of cases.

Monadic transpose. The expression ⍉A yields the array R with the last two coordinates interchanged. For a vector V, matrix M, and three dimensional array T, the following relations hold:

⍉V is equivalent to 1⍉V (and hence to V)
⍉M is equivalent to 2 1⍉M (ordinary matrix transpose)
⍉T is equivalent to 1 3 2⍉T

Rotate. If K is a scalar or one-element vector and X is a vector, then K⌽X is a cyclic rotation of X defined as follows: K⌽X is equal to X[1+(⍴X)|¯1+K+⍳⍴X] For example, if X←2 3 5 7 11, then 2⌽X is equal to 5 7 11 2 3, and ¯2⌽X is equal to 7 11 2 3 5. In 0-origin indexing, the definition for K⌽X becomes X[(⍴X)|K+⍳⍴X].

If the rank of X exceeds 1, then the coordinate J along which rotation is to be performed may be specified in the form Z←K⌽[J]X. Moreover, the dimension of K must equal the remaining dimensions of X, and each vector along the Jth coordinate of X is rotated as specified by the corresponding element of K. A scalar K is extended in the usual manner.

| Case | ⍴R | Definition |
|---|---|---|
| R←1⍉V | ⍴V | R←V |
| R←1 2⍉M | ⍴M | R←M |
| R←2 1⍉M | (⍴M)[2 1] | R[I;J]←M[J;I] |
| R←1 1⍉M | ⌊/⍴M | R[I]←M[I;I] |
| R←1 2 3⍉T | ⍴T | R←T |
| R←1 3 2⍉T | (⍴T)[1 3 2] | R[I;J;K]←T[I;K;J] |
| R←2 3 1⍉T | (⍴T)[3 1 2] | R[I;J;K]←T[J;K;I] |
| R←3 1 2⍉T | (⍴T)[2 3 1] | R[I;J;K]←T[K;I;J] |
| R←1 1 2⍉T | (⌊/(⍴T)[1 2]),(⍴T)[3] | R[I;J]←T[I;I;J] |
| R←1 2 1⍉T | (⌊/(⍴T)[1 3]),(⍴T)[2] | R[I;J]←T[I;J;I] |
| R←2 1 1⍉T | (⌊/(⍴T)[2 3]),(⍴T)[1] | R[I;J]←T[J;I;I] |
| R←1 1 1⍉T | ⌊/⍴T | R[I]←T[I;I;I] |

Table 3.9: TRANSPOSITION

For example, if ⍴X is 3 4 and J is then K must be of dimension and [J;] is equal to [ ]⌽X[I;] If J is 1, then ⍴K must be , and [;I] is equal to K[ ]⌽X[;J]. If is a three-dimensional array, then must be a matrix or a scalar. For example:

M

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |

0 1 2 3⌽[1]M

| | | | |
|---|---|---|---|
| 1 | 6 | 11 | 4 |
| 5 | 10 | 3 | 8 |
| 9 | 2 | 7 | 12 |

1 2 3⌽[2]M

| | | | |
|---|---|---|---|
| 2 | 3 | 4 | 1 |
| 7 | 8 | 5 | 6 |
| 12 | 9 | 10 | 11 |

The expression K⊖X denotes rotation along the first coordinate of X. The symbol ⊖ is formed by overstriking a ○ with a minus sign.

Reverse. If X is a vector and R←⌽X, then R is equal to X except that the elements appear in reverse order. Formally, R is equal to X[1+(⍴X)-⍳⍴X]. In 0-origin indexing, the appropriate expression is X[¯1+(⍴X)-⍳⍴X].

If A is any array, J is a scalar or one-element vector, and R←⌽[J]A, then R is an array like A except that the order of the elements is reversed along the Jth coordinate. For example:

A

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |

⌽[1]A

| | | |
|---|---|---|
| 4 | 5 | 6 |
| 1 | 2 | 3 |

⌽[2]A

| | | |
|---|---|---|
| 3 | 2 | 1 |
| 6 | 5 | 4 |

The expression ⌽A denotes reversal along the last coordinate of A, and ⊖A denotes reversal along the first coordinate. For example, if A is of rank 3, then ⌽A is equivalent to ⌽[3]A, and ⊖A is equivalent to ⌽[1]A.

Compress. The expression U/X denotes compression of X by U. If U is a logical vector (comprising elements having only the values 0 or 1) and X is a vector of the same dimension, then U/X produces a vector result of +/U elements chosen from those elements of X corresponding to non-zero elements of U. For example, if X←2 3 5 7 11 and U←1 0 1 1 0 then U/X is 2 5 7 and (~U)/X is 3 11

To be conformable, the dimensions of the arguments must agree, except that a scalar (or one-element vector) left argument is extended to apply to all elements of the right argument. Hence 1/X is equal to X and 0/X is an empty vector. A scalar right argument is not extended. The result in every case is a vector.

If $M$ is a matrix, then $U/[1]M$ denotes compression along the first coordinate, that is, the compression operates on each column vector and therefore deletes certain rows. It is called column compression. Similarly, $U/[2]M$ (or simply $U/M$) denotes row compression. The result in every case is a matrix. As in reduction, $L/M$ denotes compression along the last coordinate, and $1\neq M$ denotes compression along the first.

Expand. Expansion is the converse of compression and is denoted by $U\backslash X$. If $Y\leftarrow U\backslash X$, then $U/Y$ is equal to $X$ and (if $X$ is an array of numbers) $(\sim U)/Y$ is an array of zeros. In other words, $U\backslash X$ expands $X$ to the format indicated by the ones in $U$ and fills in zeros elsewhere. To be conformable, $+/U$ must equal $\rho X$.

If $X$ is an array of characters, then spaces are supplied rather than zeros, i.e., if $Y\leftarrow U\backslash X$ then $(\sim U)/Y$ is an array of the space character ' '. Again, $U\backslash[J]M$ denotes expansion along the $J$th coordinate, $U\backslash M$ denotes expansion along the last, and $U\backslash M$ denotes expansion along the first. See Table 3.8 for examples of expansion.

A scalar left argument is not extended.

Decode. The expression $R\perp X$ denotes the value of the vector $X$ evaluated in a number system with radices $R[1],R[2],\ldots,R[\rho R]$. For example, if $R\leftarrow 24\ 60\ 60$ and $X\leftarrow 1\ 2\ 3$ is a vector of elapsed time in hours, minutes, and seconds, then $R\perp X$ has the value 3723, and is the corresponding elapsed time in seconds. Similarly, $10\ 10\ 10\perp 1\ 7\ 6$ is equal to 176, and $2\ 2\ 2\ 2\perp 1\ 0\ 1$ is equal to 5. Formally, $R\perp X$ is equal to $+/W\times X$, where $W$ is the weighting vector determined as follows: $W[\rho W]$ is equal to 1, and $W[I-1]$ is equal to $R[I]\times W[I]$. For example, if $R$ is $2\ 3\ 4$, then $W$ is $12\ 4\ 1$.

The result is a scalar.

The arguments $R$ and $X$ must be of the same dimension, except that either may be a scalar (or one-element vector). For example, $10\perp 1\ 7\ 6$ is equal to $176$. The arguments are not restricted to integer values. If $X$ is a scalar, then $R\perp X$ is the value of a polynomial in $X$ with coefficients $R$, arranged in order of descending powers of $X$.

The decode function is commonly applied in work with fixed-base number systems and is often called the base value function.

Encode. The encode function $R\top N$ denotes the representation of the scalar $N$ in the base-$R$ number system. Thus, if $Z\leftarrow R\top N$, then $(\times/R)\mid N-R\perp Z$ is equal to zero. For example, $2\ 2\ 2\ 2\top 5$ is $0\ 1\ 0\ 1$ and $2\ 2\ 2\top 5$ is $1\ 0\ 1$ and $2\ 2\top 5$ is $2\ 1$. The dimension of $R\top N$ is the dimension of $R$. The encode function is also called representation.

Index of. If $V$ is a vector and $S$ is a scalar, then $J\leftarrow V\iota S$ yields the position of the earliest occurrence of $S$ in $V$. If $S$ does not equal any element of $V$, then $J$ has the value $(\iota 1)+\rho V$. Clearly, this value depends, as does any result of this function, on the index origin, and is one greater than the largest permissible index of $V$.

If $S$ is a vector, then $J$ is a vector such that $V[J]$ is the index in $V$ of $S[I]$. For example:

```
      'ABCDEFGH'ι'CAFFE'
7  1  6  6  5
```

If $X$ is a numerical vector, then the expression $X\iota\lceil/X$ yields the index of the (first) maximum element in $X$. For example, if $X$ is the vector $6\ 3\ 5\ 13\ 2\ 7\ 9$, then $\lceil/X$ is $13$ and $X\iota\lceil/X$ is $4$.

The result in every case has the same dimensions as the righthand argument of $\iota$. For example, if $Z\leftarrow V\iota S$, and $S$ is a matrix, then $Z[I;J]$ is equal to $V\iota S[I;J]$.

Membership. The function $X\in Y$ yields a logical array of the same dimension as $X$. Any particular element of $X\in Y$ has the value 1 if the corresponding element of $X$ belongs to $Y$, that is, if it occurs as some element of $Y$. For example, $(\iota 5)\in 3\ 5$ is equal to $0\ 1\ 0\ 1\ 0$ and $'\ldots'\in'\ldots'$ equals $0\ 0\ 1\ 0\ 1\ 1\ 0\ 0$.

If the vector $Y$ represents the universal set in some finite universe of discourse, then $X\in Y$ is the characteristic of the set $X$, and the membership function is therefore also called the characteristic function.

The size of the result of the function $\iota$ is determined by the size of the left argument, whereas the size of the result of the dyadic function $\in$ is determined by the size of the right argument. However, the left arguments of both frequently play the role of specifying the universe of discourse.

Take and drop. If $V$ is a vector and $S$ is a scalar between 0 and $\rho V$, then $S\uparrow V$ takes the first $S$ components of $V$. For example, if $V\leftarrow\iota 7$, then $3\uparrow V$ is 1 2 3 and $0\uparrow V$ is $\iota 0$, and $8\uparrow V$ yields a domain error.

If $S$ is chosen from the set $-\iota\rho V$, then $S\uparrow V$ takes the last $|S$ elements of $V$. For example, $^{-}3\uparrow V$ is 5 6 7.

If $A$ is an array, then $W\uparrow A$ is valid only if $W$ has one element for each dimension of $A$, and $W[I]$ determines what is to be taken along the $I$th coordinate of $A$. For example, if $A\leftarrow 3\ 4\rho\iota 12$, then $2\ ^{-}3\uparrow A$ is the matrix

```
2  3  4
6  7  8
```

The function drop ($\downarrow$) is defined analogously, except that the indicated number of elements are dropped rather than taken. For example, $^{-}1\ 1\downarrow A$ is the same matrix as the one displayed in the preceding paragraph.

The rank of the result of the take and drop functions is the same as the rank of the right argument. The take and drop functions are similar to the transpose in that the left argument concerns the dimension vector of the right argument.

Grade up and down. The function $\grave{A}V$ produces the permutation which would order $V$, that is $V[\grave{A}V]$ is in ascending order. For example, if $V$ is the vector 7 1 16 5 3 9, then $\grave{A}V$ is the vector 2 5 4 1 6 3, since 2 is the index of the first in rank, 5 is the index of the second in rank, and so on. The symbol $\grave{A}$ is formed by overstriking $|$ and $\Delta$.

If $P$ is a permutation vector, then $\grave{A}P$ is the permutation inverse to $P$. If a vector $D$ contains duplicate elements, then the ranking among any set of equal elements is determined by their positions in $D$. For example, $\grave{A}5\ 3\ 7\ 3\ 9\ 2$ is the vector 6 2 4 1 3 5.

The right argument of $\grave{A}$ may be any array $A$ of rank greater than zero, and the coordinate $J$ along which the grading is to be applied may be indicated by the usual notation $\grave{A}[J]A$. The form $\grave{A}A$ applies as usual to the last coordinate. The result of $\grave{A}A$ is of the same dimension as $A$.

The grade down function $\forall$ is the same as the function $\grave{A}$ except that the grading is determined in descending order. Because of the treatment of duplicate items, the expression $\wedge/(\grave{A}V)=\phi\forall V$ has the value 1 if and only if the elements of the vector $V$ are all distinct.



Deal. The function $M?N$ produces a vector of dimension $M$ obtained by making $M$ random selections, without replacement, from the population $\iota N$. In particular, $N?N$ yields a random permutation of order $N$. Both arguments are limited to scalars or one-element arrays.

Comments. The lamp symbol $\grave{a}$, formed by overstriking $\cap$ and $\circ$, signifies that what follows it is a comment, for illumination only and not to be executed; it may occur only as the first character in a statement, but may be used in defined functions.

## MULTIPLE SPECIFICATION

Specification ($\leftarrow$) may (like any other function) occur repeatedly in a single statement. For example, the execution of the statement $Z\leftarrow X\times A\leftarrow 3$ will assign to $A$ the value 3, then multiply this assigned value of $A$ by $X$ and assign the resulting value to $Z$.

Multiple specification is useful for initializing variables. For example:

$$X\leftarrow Y\leftarrow 1+Z\leftarrow 0$$

sets $X$ and $Y$ to 1 and $Z$ to 0.

A branch may occur in a statement together with one or more specifications, provided that the branch is the last operation to be executed (i.e., the leftmost). For example, the statement $\rightarrow S\times\iota N>I\leftarrow I+1$ first augments $I$, and then branches to statement $S$ if $N$ exceeds the new value of $I$.

In the expression $Z\leftarrow(A+B)\times(C+D)$ it is immaterial whether the left or the right argument of the $\times$ is evaluated first, and hence no order is specified. The principle of no specified order in such cases is also applied when the expressions include specification. Since the order here is sometimes material, there is no guarantee which of two or more possible results will be produced.

Suppose, for example, that $A$ is assigned the value 5 and the expression $Z\leftarrow(A\leftarrow 3)\times A$ is then executed. If the left argument of $\times$ is executed first, then $A$ is assigned the value 3, the right argument then has the new value 3 and $Z$ is finally assigned the value 9. If, on the other hand, the right argument is evaluated first it has the value 5 initially assigned to $A$, the value 3 is then assigned to $A$ and multiplied by the 5 to yield a value of 15 to be assigned to $Z$.

SYSTEM DEPENDENT FUNCTIONS

There are three main types of information about the state of the system which are of value to the user:

1. general information common to all users, such as date, time of day, and the current number of terminals connected to the system.

2. information specific to the particular work session, such as the time of sign-on, the central computer time used, and the total keying time.

3. information specific to the active workspace, such as the amount of storage available, and the condition of the state indicator.

This information is provided by a single family of functions denoted by ⌶ (formed by overstriking ⍳, and ⊥), and called the I-Beam functions. The individual member function is selected by the argument as shown in Table 3.10. Times are all in units of one-sixtieth of a second, the date is given as a six-digit integer in which the successive digit pairs specify the month, day, and year, and the available storage is given in bytes.

The byte is a unit of storage equal to 8 binary digits. A variable requires for storage a small number of bytes of overhead, plus a certain number of bytes per element depending upon the form of its representation: 1 if the elements are characters, 0.125 if the elements are logical, 4 if the elements are integers less than ¯32 in magnitude, and 8 for other numbers.

In designing an algorithm for a particular purpose, it frequently happens that one may trade time for space; that is, an algorithm which requires little computer time may require more storage space for intermediate results, and an algorithm which requires little storage may be less efficient in terms of time. Hence, the information provided by the functions ⌶9 (computer time used) and ⌶22 (available storage space) may be helpful in designing algorithms. For example, the function TIME of Appendix B can be used to determine the computer time used in the execution of a function.

Moreover, since the functions ⌶9 and ⌶22 can, like all of the I-beam functions, be used within a defined function, they can be used to make the execution dependent upon the space available or the computer time used.

| X | Definition of ⌶X |
|---|---|
| 19 | Accumulated keying time (time during which the keyboard has been unlocked awaiting entries) during this session. |
| 20 | The time of day. |
| 21 | The central computer time used in this session. |
| 22 | The amount of available space (in bytes). |
| 23 | The number of terminals currently connected. |
| 24 | The time at the beginning of this session. |
| 25 | The date. |
| 26 | The first element of the vector ⌶27. |
| 27 | The vector of statement numbers in the state indicator. |

NOTES
1. All times in 1/60 seconds
2. Date is represented by a 6-digit integer; successive digit pairs represent month, day, and year.
3. ⌶27 yields a vector; all other results are scalars.

Table 3.10: SYSTEM DEPENDENT FUNCTIONS

Keying time is defined as the total accumulated time since sign-on during which the keyboard has been unlocked awaiting entry. The associated function (⌶19) may be used in conjunction with ⍞ or ⎕ input to determine the amount of time taken by a student in responding to a question. The following example shows the definition and use of a multiplication drill which tells the student how long he has taken (in whole minutes and seconds) to answer each question:

```
      ∇MULTDRILL N;X;Y;TIME
[1]   ⎕←X←?N
[2]   TIME←⌶19
[3]   Y←⎕
[4]   TIME←(⌶19)-TIME
[5]   →8×1X≠×/Y
[6]   'TIME: ';2↑60 60 ⊤ TIME
[7]   →1
[8]   'NO, TRY AGAIN'
[9]   →3
      ∇
      MULTDRILL 12 12
 
      :
      
      
      
```

Such a drill could be expanded to accumulate statistics of the student's response times or to use some function of the response times to control the difficulty of the questions posed.

Since times are expressed in units of 1÷60 seconds, the time in hours, minutes, and seconds can be determined by an expression of the form 3↑24 60 60 60⊤⌶21. Similarly, a 3-element vector representing the date can be obtained from the expression (3⍴100)⊤⌶25.

The expression ⌶27 provides the vector of statement numbers in the state indicator, with the first position occupied by the number of the statement on which the innermost function is suspended. If no functions are suspended, the vector ⌶27 is empty.

The expression ⌶26 yields a scalar which is the first element of ⌶27. It is therefore equal to the number of the statement being, or about to be, executed and is particularly useful in branches. For example, →*N*+⌶26 causes a forward jump of *N* statements. Moreover, entering →⌶26 is a safe way to resume execution without having to read and enter the statement number printed at the point of the last suspension. It is even more convenient to resume by entering →*C*, after first defining the function *C* as follows:

```
     ∇Z←C
[1]  Z←(⌶27)[2]∇
```



## PART 4

## LIBRARY FUNCTIONS

A user may load or copy functions from any workspace for which he knows the library number and workspace name (and password, if any). Moreover a listing of the workspaces in Library *N* can be obtained by the command )*LIB N* for any public library, i.e., for any library whose number is below 1000.

A public library may be used for the casual sharing of functions among a group of co-workers. When intended for more general use, a library function should be thoroughly tested and well-documented, and should incorporate messages for the guidance of the user. It is therefore good practice to restrict certain of the public libraries to such functions as are of general interest and have passed appropriate acceptance tests.

In the APL\360 system as distributed, Library 1 is restricted in this manner. This section treats each of the workspaces in this library by loading each and displaying the descriptions contained in the workspaces themselves. Further information on the functions in each workspace can (except in the case of the locked functions in *WSFNS*) be obtained by displaying the function definitions.

```
      )LOAD 1 ADVANCEDEX
ADVANCEDEX SAVED 07/14/68   16.53.19
      )FNS
AH      ASSOC   BIN     COMB    DTH     ENTER   F       FC
GC      GCD     GCV     HILB    HTD     IN      INV     INVP
IN1     LFC     LOOKUP  PALL    PER     PERM    PO      POL
POLY    POLYB   RESET   TIME    TRUTH   ZERO
      DESCRIBE

EACH OF THE VARIABLES OF THIS WORKSPACE WHICH BEGINS WITH THE
LETTER D IS THE DESCRIPTION OF THE FUNCTION WHOSE NAME IS
OBTAINED BY REMOVING THE D. FOR FURTHER DETAILS SEE APPENDIX
B OF THE APL\360 MANUAL.
```

```
      )LOAD 1 PLOTFORMAT
PLOTFORMAT SAVED 07/20/68   31.07.27
      )FNS
AND     DESCRIBE        DFT     EFT     PLOT    VS

      DESCRIBE

THE FUNCTIONS INCLUDED IN THIS WORKSPACE ARE LISTED BELOW:

SYNTAX                  DESCRIPTION

Z←A AND B   ESSENTIALLY A COLUMN-CATENATOR, WITH SOME EXTRA
            EFFECTS WHEN THE ARGUMENTS ARE NOT MATRICES.
            THIS FUNCTION IS DESIGNED TO BE USED EITHER
            INDEPENDENTLY, OR IN CONJUNCTION WITH  VS.  IT
            PROVIDES A CONVENIENT WAY OF FORMING INPUT
            TO DFT AND EFT.

Z←A DFT B   FORMS FIXED-POINT OUTPUT.  MORE DETAILED DIREC-
            TIONS CAN BE FOUND IN THE VARIABLE  HOWFORMAT

Z←A EFT B   FORMS EXPONENTIAL OUTPUT.  MORE DETAILED DIREC-
            TIONS CAN BE FOUND IN THE VARIABLE  HOWFORMAT

A PLOT B    GRAPHS ONE OR MORE FUNCTIONS SIMULTANEOUSLY.
            DIRECTIONS FOR USING  PLOT  CAN BE FOUND IN THE
            VARIABLE  HOWPLOT

  ←A VS B   ESSENTIALLY A COLUMN-CATENATOR, SIMILAR TO  AND,
            EXCEPT THAT THE RIGHT-HAND ARGUMENT MUST BE OF
            RANK ≤ 1   IT IS DESIGNED PRIMARILY TO PROVIDE
            CONVENIENT FORMATION OF INPUT TO  PLOT  FUNC-
            TION. WHETHER USED BY ITSELF OR WITH  AND,  VS
            WILL CAUSE ITS RIGHT ARGUMENT TO APPEAR AS THE
            LEFTMOST COLUMN OF THE RESULTANT ARRAY.  (THE
            RESULTANT WILL BE AN ARRAY OF RANK THREE,
            CONSISTING OF A SINGLE PLANE)

THE  AND  AND  VS  WORK WITH EITHER 1 OR 0-ORIGIN INDEXING
```

```
      HOWFORMAT

THE FUNCTIONS DFT AND EFT WILL ARRAY  NUMBERS IN DECIMAL AND
EXPONENTIAL FORM,  RESPECTIVELY, FOR  TABULAR OUTPUT.  THEY
MAY BE  USED TO  GENERATE IMMEDIATE OUTPUT,  OR TO  STORE AN
IMAGE FOR LATER PRINTING.  THE TWO FORMS ARE:

      PATTERN DFT TABLE
      PATTERN EFT TABLE
AND
      IMAGE←PATTERN DFT TABLE
      IMAGE←PATTERN EFT TABLE

THESE FUNCTIONS WORK PROPERLY ONLY WITH 1-ORIGIN INDEXING

RIGHT ARGUMENT: AN ARRAY TO BE FORMED

      IT MUST  BE NUMERICAL,  AND OF  RANK ≤  3.   THE FIRST
      PLANE  OF  A 3-DIMENSIONAL ARRAY WILL BE  TREATED AS A
      MATRIX,  AND  ALL OTHER  PLANES WILL  BE DISREGARDED
      ARRAYS OF  HIGHER  RANK  WILL BE SIGNALLED AS  A 'RANK
      PROBLEM.'

LEFT ARGUMENT: ONE  OR MORE INTEGERS TO  CONTROL THE FORMAT.
      FRACTIONAL  NUMBERS WILL  BE  SIGNALLED  AS A  'DOMAIN
      PROBLEM.'

      A SINGLE INTEGER.
      DFT: SPECIFIES  THE NUMBER OF  DIGITS TO THE  RIGHT OF
      THE DECIMAL POINT IN DECIMAL FORMAT.
      EFT:  SPECIFIES THE  NUMBER OF  SIGNIFICANT DIGITS  IN
      EXPONENTIAL FORMAT   ONE DIGIT  ALWAYS APPEARS  TO THE
      LEFT OF  THE DECIMAL  POINT.   COLUMNS WILL  BE SPACED
      UNIFORMLY, WITH SPACING  SUCH  THAT THERE  WILL BE TWO
      SPACES BETWEEN THE CLOSEST  NUMBERS

      A  PAIR OF  INTEGERS:  THE FIRST  SPECIFIES THE  TOTAL
      NUMBER OF  SPACES TO  BE ALLOCATED TO EACH COLUMN, AND
      THE SECOND  IS USED AS ABOVE.
       FT:  THE FIRST  NUMBER MUST  BE AT  LEAST TWO  LARGER
      THAN THE SECOND
      EFT.  THE FIRST  NUMBER MUST  BE AT  LEAST SIX  LARGER
      THAN THE  SECOND    IF THE LEFT  NUMBER IS  TOO SMALL,
      THIS WILL BE SIGNALLED AS A 'DOMAIN PROBLEM.'

      MORE THAN ONE PAIR OF INTEGERS: THERE MUST BE ONE PAIR
      FOR EACH COLUMN  OF OUTPUT (OR EACH ELEMENT  OF A VEC-
      TOR)    EACH PAIR  WILL BE  INTERPRETED AS  ABOVE, AND
      WILL  APPLY TO THE LAYOUT OF THE CORRESPONDING ITEMS.
      IF  THE  NUMBER OF PAIRS DOES NOT MATCH THE NUMBER OF
      COLUMNS, THIS WILL BE SIGNALLED AS A 'LENGTH ERROR.'
```

HOWPLOT

THE FUNCTION PLOT WILL GRAPH ONE OR MORE FUNCTIONS SIMULTANEOUSLY, AUTOMATICALLY SCALING THE VALUES TO FIT APPROXIMATELY WITHIN SCALE DIMENSIONS SPECIFIED BY THE USER. IT WILL WORK ONLY IN 1-ORIGIN INDEXING.

THE FORM IN WHICH PLOT IS USED IS.

SCALESIZE PLOT FUNCTION

LEFT ARGUMENT: ONE OR TWO NUMBERS.

THE FIRST NUMBER SPECIFIES THE APPROXIMATE SIZE OF THE VERTICAL AXIS AND THE SECOND NUMBER DOES THE SAME FOR THE HORIZONTAL AXIS.

IF ONLY ONE NUMBER IS SUPPLIED, IT IS APPLIED TO BOTH AXES.

THERE IS NO BUILT-IN LIMIT TO THE DIMENSIONS, AND A HORIZONTAL AXIS LARGER THAN THE WORKSPACE WIDTH WILL CAUSE SOME POINTS TO BE PRINTED ON THE NEXT LOWER LINE

RIGHT ARGUMENT: A RECTANGULAR ARRAY WITH RANK ≤ 3.

SCALAR: WILL BE TREATED AS A VECTOR OF LENGTH ONE.

VECTOR: WILL BE PLOTTED AS ORDINATE AGAINST ITS OWN INDICES AS ABSCISSA.

MATRIX: THE LEFTMOST COLUMN WILL BE TAKEN AS THE ABSCISSA AND ALL OTHER COLUMNS WILL BE PLOTTED AS ORDINATES. A DIFFERENT PLOTTING SYMBOL UP TO THE NUMBER OF SYMBOLS AVAILABLE WILL BE USED FOR EACH COLUMN. IN CASE TWO ORDINATES HAVE A COMMON POINT, THE SYMBOL FOR THE COLUMN FURTHEST TO THE RIGHT WILL BE USED.

3-DIMENSIONAL ARRAY: THE FIRST PLANE WILL BE PLOTTED AS A MATRIX, AND ALL OTHER PLANES WILL BE DISREGARDED.

AUXILIARY FUNCTIONS: THE FUNCTIONS AND ANY VS CAN BE USED TO GENERATE THE RIGHT ARGUMENT IN THE PROPER FORM FOR PLOT. FOR EXAMPLE.

Y PLOT X AND Y VS X

PLOT CHARACTERS: THE SYMBOLS USED ARE ASSIGNED TO THE VARIABLE PC IN LINE 1 OF PLOT. THE ALPHABET SUPPLIED IS 'O*•⍟Δ□' THIS ALPHABET MAY BE EXTENDED AND MODIFIED AS DESIRED, USING THE NORMAL FUNCTION-EDITING PROCEDURES: EITHER CHANGE LINE 1 OF THE FUNCTION, OR DELETE IT AND INDEPENDENTLY SPECIFY A VALUE FOR PC.

HISTOGRAMS: PLOT CAN BE USED TO GENERATE HISTOGRAMS BY SETTING THE VARIABLE HS TO 1 IN LINE 2 OF THE FUNCTION. ALTERNATIVELY, LINE 2 CAN BE DELETED, AND HS CAN BE SET EXTERNALLY.

```
    )LOAD 1 APLCOURSE
APLCOURSE SAVED 07/19/68   25.58.06
      )FNS
B1X      CHECK    DESCRIBE        DIM      DRILL    DYAD1    DYAD2
EASY     EASYDRILL        FORM    FUNDRILL         GET      INPUT
INTER    LOG      QUES    RANDOM  REDSCAPATCH      REPP
SETPARAMETERS     TEACH    TRACE
```

DESCRIBE

THE MAIN FUNCTIONS IN THIS LIBRARY WORKSPACE ARE:

TEACH
EASYDRILL

ALL OTHER FUNCTIONS ARE SUBFUNCTIONS AND ARE NOT SELF-CONTAINED.

| SYNTAX | DESCRIPTION |
|---|---|
| TEACH | AN EXERCISE IN APL FUNCTIONS USING SCALARS AND VECTORS. THE FUNCTION PRINTS OUT THE CHOICES AND OPTIONS AVAILABLE. EXAMPLES ARE SELECTED AT RANDOM WITH A RANDOM STARTING POINT. |
| EASYDRILL | THIS IS THE SAME AS TEACH EXCEPT THAT THE PROBLEMS SELECTED ARE GENERALLY SIMPLER IN STRUCTURE. PROBLEMS INVOLVING VECTORS OF LENGTH ZERO OR ONE ARE EXCLUDED. |

```
NOTE:  FOR EITHER FUNCTION, A RESPONSE OF - PLEASE - WILL
DISCLOSE THE PROPER ANSWER.  A RESPONSE OF - STOP - WILL
TERMINATE THE DRILL.

      TEACH
ARE YOU ALREADY FAMILIAR WITH THE INSTRUCTIONS? (TYPE
Y FOR YES AND N FOR NO.)
N

THIS  IS AN  EXERCISE IN  SIMPLE APL  EXPRESSIONS. YOU  WILL
FIRST HAVE THE  OPPORTUNITY TO SELECT THE  FEATURES YOU WISH
TO  BE DRILLED  IN.  THE EXERCISE  THEN BEGINS.  FOR  EACH
PROBLEM  YOU MUST  ENTER THE  PROPER RESULT.  ANSWERS  WILL
CONSIST OF SCALAR INTEGERS IF EXERCISES WITH VECTORS ARE NOT
DESIRED;  OTHERWISE  ANSWERS  WILL  CONSIST  OF  SCALARS  OR
VECTORS.  A VECTOR OF LENGTH ZERO  REQUIRES THE RESPONSE ⍬,
A VECTOR OF  LENGTH ONE REQUIRES THE RESPONSE X,  WHERE X IS
THE VALUE  OF THE ELEMENT.  YOU  HAVE THREE TRIES  FOR EACH
PROBLEM.  TYPE  STOP  AT ANY TIME TO TERMINATE THE EXERCISE
AND  PRODUCE  A  RECORDING  OF  YOUR  PERFORMANCE.  TYPING
STOPSHORT  WILL TERMINATE THE EXERCISE BUT WILL NOT  PRODUCE
A RECORD OF  PERFORMANCE.  TYPING  PLEASE  FOR ANY PROBLEM
WILL  LET YOU  PEEK AT  THE  ANSWERS. TYPE  Y  UNDER  EACH
FUNCTION FOR WHICH YOU WANT EXERCISE.


SCALAR DYADIC FUNCTIONS
+-×÷⋆⌈⌊<≤=≥>≠!|∧∨●⍲⍱
YYYYY          Y
SCALAR MONADIC FUNCTIONS
+-×÷⌈⌊!|~
    YY
TYPE Y IF EXERCISES ARE TO USE VECTORS, N OTHERWISE
N
                    ¯6×¯3
□:
      18
                    ⌊¯2.5
□:
      ¯2
TRY AGAIN
□:
      ¯3
                    0+7
□:
      STOPSHORT
```

```
      )LOAD 1 WSFNS
WSFNS SAVED 07/20/68   31.25.23

      )FNS
DELAY   DESCRIBE        DIGITS  ORIGIN  SETLINK SFEI    WIDTH

      DESCRIBE

     THE  FUNCTIONS  ORIGIN,  WIDTH,  AND  DIGITS  ARE  EACH
SIMILAR TO THE COMMAND OF THE SAME NAME, EXCEPT THAT EACH IS
A FUNCTION RATHER  THAN A COMMAND AND MAY  THEREFORE BE USED
WITHIN OTHER FUNCTIONS.  EACH HAS AN  EXPLICIT RESULT WHICH
IS THE PREVIOUS VALUE OF THE RELEVANT SYSTEM PARAMETER.

     FOR EXAMPLE, THE FOLLOWING FUNCTION:

      ∇F X
[1]   X←ORIGIN X
[2]   G
[3]   X←ORIGIN X
∇

WILL EXECUTE  THE FUNCTION G  WITH WHATEVER INDEX  ORIGIN IS
SPECIFIED BY THE  ARGUMENT OF F, AND WILL  RESTORE THE INDEX
ORIGIN TO THE VALUE THAT IT HAD BEFORE THE EXECUTION OF F.

     THE FOLLOWING FUNCTIONS ARE ALL  AVAILABLE:

SYNTAX          DESCRIPTION

Z←SETLINK X     SETS  THE VALUE  OF THE LINK  IN THE  CHAIN OF
                NUMBERS GENERATED  IN THE USE OF  THE ROLL AND
                DEAL FUNCTIONS.  THE EXPLICIT RESULT PRODUCED
                BY SETLINK IS THE PREVIOUS VALUE OF THE LINK.

                THE  RESULTS PRODUCED  BY  THE  ROLL AND  DEAL
                FUNCTIONS  ARE NOT  THE LINKS  THEMSELVES, BUT
                RATHER SOME  FUNCTION OF THEM.  THE LENGTH OF
                THE CHAIN (BEFORE REPETITION) IS ¯1+2*31.

DELAY X         DELAYS EXECUTION FOR X SECONDS

SFEI X          SETS  THE  SIGNAL  MESSAGE FOR  EVALUATED INPUT
                (SEE THE  SECTION ON INPUT AND  OUTPUT, PART 1
                OF THE  APL\360  USER'S MANUAL).  THE ARGUMENT
                MUST BE A LINE OF NO MORE THAN 7 CHARACTERS.
```

```
      )LOAD 1 TYPEDRILL
TYPEDRILL SAVED 07/14/68   19.42.16

      )FNS
DESCRIBE        IN      INSTRUCTIONS    MATCH   PRT     QUERY
STATISTICS      TIME    TYPEDRILL       WS
```

*DESCRIBE*

*THE MAIN FUNCTION IN THIS WORKSPACE IS TYPEDRILL; ALL OTHERS ARE SUBFUNCTIONS. TO USE IT, SIMPLY ENTER*

*TYPEDRILL*

*TYPEDRILL IS A TIMED TYPING EXERCISE. THE SYSTEM RESPONDS WITH THE STATEMENT 'YOU ARE IN CONTROL STATE'. FOUR COMMANDS ARE AT YOUR DISPOSAL: ENTER, DRILL, STAT, AND STOP. ENTERING ONE OF THEM BRINGS YOU INTO THAT STATE:*

*ENTER: YOU MAY ENTER ONE-LINE SENTENCES OR EXPRESSIONS ON WHICH YOU WISH TO BE DRILLED. ENTERING A BLANK LINE (CARRIAGE RETURN ONLY) RETURNS YOU TO THE CONTROL STATE.*

*DRILL: ONE OF THE LINES ENTERED VIA THE ENTER STATE IS SELECTED AT RANDOM AND PRINTED. YOU ARE THEN EXPECTED TO ENTER THE SAME LINE. IF IT IS CORRECT, THE TIME TAKEN IS PRINTED (IN SECONDS), IF NOT YOU ARE ASKED TO RETYPE IT. A BLANK LINE CAUSES RETURN TO THE CONTROL STATE.*

*STAT: THE ACCUMULATED STATISTICS ARE PRINTED. THE HORIZONTAL AXIS SHOWS THE TRIAL NUMBERS AND THE VERTICAL SHOWS THE TIME IN SECONDS. A VERTICAL ARROW INDICATES THAT THE TIME EXCEEDED THE LIMITS OF THE GRAPH. THE RETURN TO THE CONTROL STATE IS AUTOMATIC.*

*STOP: STOPS THE DRILL AND PRINTS THE STATISTICS.*

```
      TYPEDRILL
CONTROL WORDS ARE: ENTER, DRILL, STAT, AND STOP

YOU ARE IN CONTROL STATE
ENTER
NOW IS THE TIME FOR ALL GOOD MEN TO COME TO
I SING OF OLAF GLAD AND BIG
X←|⌈P×Q+Y*R≤5

YOU ARE IN CONTROL STATE
DRILL
NOW IS THE TIME FOR ALL GOOD MEN TO COME TO
NOW IS THE TIME OFR ALL GOOD MEN TO COME TO
                ^
NOW IS THE TIME FOR ALL GOOD MEN TO COME TO
16.9
X←|⌈P×Q+Y*R≤5
X←|⌈P×Q+Y*R≤5
19.9
I SING OF OLAF GLAD AND BIG

YOU ARE IN CONTROL STATE
STOP
```

## Appendix A

## SAMPLE TERMINAL SESSION

```
)1776
010 )  19.32.36 07/03/68  JANET

   A P L \ 3 6 0
```

FUNDAMENTALS

```
      3×4
12
      X←3×4
```

Entry automatically indented
Response not indented
X is assigned value of the expression

```
      X
12
      Y←¯5
```

Value of X typed out
Negative sign for negative constants

```
      X+Y
7
      144E¯2
1 44
```

Exponential form of constant

```
      P←1 2 3 4
      P×P
1  4  9  16
      P×Y
¯5  ¯10  ¯15  ¯20
      Q←'CATS'
      Q
CATS
```

Four-element vector
Functions apply element by element
Scalar applies to all elements
Character constant (4-element vector)

```
      YZ←5
      YZ[1]←5
      YZ+YZ[1]
10
```

Multi-character names

```
      3+4×5+6
         ∨
         +5+6
18
      X←3
      Y←4
      (X×Y)+4
16
      X×Y+4
24
```

Correction by backspace and linefeed
Executed from right to left

```
      X Y
SYNTAX ERROR
      X Y
        ^
      XY
VALUE ERROR
      XY
        ^
```

Entry of invalid expression
Shows type of error committed
Retypes invalid statement with caret where execution stopped
Multi-character name (not X×Y)

XY had not been assigned a value

SCALAR FUNCTIONS

```
      4×3⌈5.1
20.4
      (4×3)⌈5.1
12
      4×⌈5.1
24
```

Dyadic maximum

Monadic ceiling

```
      X←⍳5
      X
1  2  3  4  5
      ⍳0
```

Index generator function

Empty vector
prints as a blank line

```
      Y←5-X
      Y
4  3  2  1  0
      X⌈Y
4  3  3  4  5
      X≤Y
1  1  0  0  0
      ○1
3.141592654
      ○1 2
3.141592654  1.570796327
      X←45 90
      ○X÷180
0.7853981634  1.570796327
      1○1
0.8418709848
      2○1 2
0.5403023059  ¯0.4161468365
      3○1
1.557407725
      ¯3○1
0.7853981634
      3○¯3○⍳7
1  2  3  4  5  6  7
      Y←1 2
      4○Y
1.414213562  2.236067977
      0○÷Y
0  0.8660254038
      7○1 2
0.761594156  0.9640275801
      ¯7○7○1 2
1  2
```

All scalar functions extend to vectors

Relations produce logical (0 1) results

Pi×1

Pi÷1 2

Conversion of X to radians

Sin 1

Cos 1 2

Tan 1

Arctan 1

Tan Arctan 1 2 3 4 5 6 7

(1+Y*2)*.5

(1-÷Y*2)*.5

Tanh 1 2

Arctanh Tanh 1 2

## DEFINED FUNCTIONS

```
      ∇Z←X F Y                    Header (2 args and result)
[1]   Z←((X*2)+Y*2)*.5            Function body
[2]   ∇                           Close of definition
      3 F 4                       Execution of dyadic function F
5
      P←7
      Q←(P+1)F P-1                Use of F with expressions
      Q                                as arguments
10
      4×3 F 4
20
      ∇B←G A                      G is the signum function
[1]   B←(A>0)-A<0                 A and B are local variables
[2]   ∇
      G 4
1
      G ¯6
¯1
      X←¯6
      G X
¯1
      ∇H A                        Like G but has no explicit result
[1]   P←(A>0)-A<0                 P is a global variable
[2]   ∇
      H¯6
      P
¯1
      Y←H¯6                       H has no explicit result
VALUE ERROR                            and hence produces a value
      Y←H ¯6                           error when used to right
        ∧                              of assignment
      ∇Z←FAC N;I                  FAC is the factorial function
[1]   Z←1
[2]   I←0
[3]   L1:I←I+1                   L1 becomes 3 at close of def
[4]   →0×ιI>N                    Branch to 0 (out) or to next
[5]   Z←Z×I
[6]   →L1                         Branch to L1 (that is, 3)
[7]   ∇
      FAC 3
6
      FAC 5
120
      TΔFAC←3 5                   Set trace on lines 3 and 5 of FAC
      X←FAC 3
FAC[3] 1                          Trace of FAC
FAC[5] 1
FAC[3] 2
FAC[5] 2
FAC[3] 3
FAC[5] 6
FAC[3] 4
      TΔFAC←0                     Reset trace control
```

## MECHANICS OF FUNCTION DEFINITION

```
      ∇G←M GCD N
[1]   G←N                         Greatest common divisor
[2]   M←M|N                             function based on the
[3]   →4×M≠0                            Euclidean algorithm
[4]   [1]G←M
[2]   [4]N←G                      Correction of line 1
[5]   [1□]                        Resume with line 4
[1]   G←M                         Display line 1
[1]   [□]
    ∇ G←M GCD N                   Display entire GCD Function
[1]   G←M
[2]   M←M|N
[3]   →4×M≠0
[4]   N←G
    ∇
[5]   →1                          Close of display, not close of def
[6]   ∇                           Enter line 5
      36 GCD 44                   Close of definition
4                                 Use of GCD
      ∇GCD                        4 is GCD of 36 and 44
[6]   [4.1]M,N                    Reopen def (Use ∇ and name only)
[4.2] [□]                         Insert between 4 and 5
    ∇ G←M GCD N                   Display entire function
[1]   G←M
[2]   M←M|N
[3]   →4×M≠0
[4]   N←G
[4.1] M,N                         Fraction stays until close of def
[5]   →1
    ∇                             End of display
[6]   ∇                           Close of definition
      36 GCD 44
8  36                             Iterations printed by
4  8                                   line 5 (was line 4.1)
4                                 Final result
      ∇GCD[□]∇                    Reopen, display, and close GCD
    ∇ G←M GCD N
[1]   G←M
[2]   M←M|N
[3]   →4×M≠0
[4]   N←G
[5]   M,N                         Line numbers have been
[6]   →1                                reassigned as integers
    ∇                             Close (Even number of ∇'s in all)
      ∇GCD                        Reopen definition of GCD
[7]   [5]                         Delete line 5 by linefeed
        ∧
      ∇                           Close definition
```

```
      ∇Z←ABC X
[1]   Z←(33×Q←(R×5)-6
[2]   [1□9]
[1]   Z←(33×Q←(R×5)-6
         /   1 /1
[1]   Z←(3×Q)+(T×5)-6
[2]   ∇
      FAC 5
120
      )ERASE FAC
      FAC 5
SYNTAX ERROR
      FAC 5
      ^
      ∇Z←BIN N
[1]   LA:Z←(Z,0)+0,Z
[2]   →LA×N≥ρZ∇
      BIN 3
VALUE ERROR
BIN[1] LA:Z←(Z,0)+0,Z
                  ^
      Z←1
      →1
1  3  3  1
      BIN 4
VALUE ERROR
BIN[1] L1:Z←(Z,0)+0,Z
                  ^
      ∇BIN[ 1]Z←1∇
      )SI
BIN[1] *
      →1
1  4  6  4  1
      ∇BIN[□]∇
    ∇ Z←BIN N
[1]   Z←1
[2]   LA:Z←(Z,0)+0,Z
[3]   →LA×N≥ρZ
    ∇
      SΔBIN←2
      Q←BIN 3

BIN[2]
      Z
1
      →2

BIN[2]
      →2

BIN[2]
      →0
```

A function to show line editing
A line to be corrected
Initiate edit of line 1
Types line, stops ball under 9
Slash deletes, digit inserts spaces
Ball stops at first new
space. Then enter ) T
FAC still defined

Erase function FAC
Function FAC no longer exists

An (erroneous) function for
binomial coefficients

Suspended execution

Assign value to Z
Resume execution
Binomial coefficients of order 3

Same error (local variable Z
does not retain its value)

Insert line to initialize Z
Display state indicator
Suspended on line 1 of BIN
Resume execution (BIN now correct)

Display revised function
and close definition

Set stop on line 2
Execute BIN

Stop due to stop control
Display current value of Z

Resume execution

Stop again on next iteration
Resume

Stop again
Branch to 0 (terminate)

```
      ∇MULTDRILL N;Y;X
[1]   Y←?N
[2]   Y
[3]   X←□
[4]   →0×ιX='S'
[5]   →ιX=×/Y
[6]   'WRONG, TRY AGAIN'
[7]   →3∇
      MULTDRILL 12 12
2  10
□:
      37
WRONG, TRY AGAIN
□:
      20
6  7
□:
      'S'
      ∇Z←ENTERTEXT
[1]   Z←''
[2]   D←ρZ
[3]   Z←Z,⍞
[4]   →2×D≠ρZ
[5]   ∇
      Q←ENTERTEXT
THIS IS ALL
 CHARACTER INPUT

      Q
THIS IS ALL CHARACTER INPUT
      N←5
      'NOTE: ι';N;' IS ';ιN
NOTE:ι5 IS 1  2  3  4  5
```

INPUT AND OUTPUT

A multiplication drill
ρN random integers
Print the random factors
Keyboard input
Stop if entry is the letter S
Repeat if entry is correct product
Prints if preceding branch fails
Branch to 3 for retry
Drill for pairs in range 1 to 12

Indicates that keyboard entry
is awaited

Entry of letter S stops drill
Example of character (⍞) input
Make Z an empty vector
D is the length of Z
Append character keyboard entry
Branch to 2 if length increased
(i.e., entry was not empty)

Keyboard
entries
Empty input to terminate
Display Q

Mixed output statement

RECTANGULAR ARRAYS

```
      P←2 3 5 7
      ρP
4
      T←'OH MY'
      ρT
5
      P,P
2  3  5  7  2  3  5  7
      T,T
OH MYOH MY
      T,P
DOMAIN ERROR
      T,P
      ^
```

Dimension of P

Character vector

Catenation

Characters cannot be catenated
with numbers

```
      M←2 3⍴2 3 5 7 11 13          Reshape to produce a 2×3 matrix
      M                            Display of an array of rank >1
                                        is preceded by a blank line
2  3  5
7 11 13
      2 4⍴T                        A 2×4 matrix of characters

OH M
YOH
      6⍴M                          A matrix reshaped to a vector
2 3 5 7 11 13
      ,M                           Elements in row-major order
2 3 5 7 11 13
      P←,M
      P[3]                         Indexing (third element of P)
5
      P[1 3 5]                     A vector index
2 5 11
      P[⍳3]                        The first three elements of P
2 3 5
      P[⍴P]                        Last element of P
13
      M[1;2]                       Element in row 1, column 2 of M
3
      M[1;]                        Row 1 of M
2 3 5
      M[1 1;3 2]                   Rows 1 and 1, columns 3 2

 5 3
 5 3
      A←'ABCDEFGHIJKLMNOPQ'        The alphabet to Q
      A[M]                         A matrix index produces
                                        a matrix result
BCE
GKM
      A[M[1 1;3 2]]

EC
EC
      M[1;]←15 3 12                Respecifying the first row of M
      M

 15  3 12
  7 11 13
```

```
      Q←3 1 5 2 4 6                A permutation vector
      P[Q]                         Permutation of P
5 2 11 3 7 13
      Q[Q]                         A new permutation
5 3 4 1 2 6
      P[3]                         Present index origin is 1
5
      )ORIGIN 0                    Set index origin to 0
WAS 1
      P[3]
7
      P[0 1 2]                     First three elements of P
2 3 5
      ⍳5                           Result of index generator
0 1 2 3 4                               begins at origin
      )ORIGIN 1
WAS 0
      ⍳5
1 2 3 4 5
```

FUNCTIONS ON ARRAYS

```
      V←?3⍴9                       Vector of 3 random integers (1-9)
      M←?3 3⍴9                     Random 3 by 3 matrix
      N←?3 3⍴9                     Random 3 by 3 matrix
      V
2 1 7
      M
 7 9 4
 5 8 1
 1 5 7
      N
 1 4 1
 4 7 6
 9 8 5
      M+N                          Sum (element-by-element)
 8 13  5
 9 15  7
10 13 12
```

```
      M⌈N
 7  9  4
 5  8  6
 9  8  7
      M≤N
0 0 0
0 0 1
1 1 0
      +/V
10
      ×/V
14
      +/[1]M
13  22  12
      +/[2]M
20  14  13
      +/M
20  14  13
      ⌈/M
9  8  7
      X←1 5
      +/(1 2○X)*2
1
      ○/1 2,X
0 07067822453
      Y←○/0 2,X
      Y
0 9974949866
      Y=1○X
1
      M+.×N
  79  123   81
  46   84   58
  84   95   66
      M+.≤N
1 1 1
1 1 1
2 3 2
      M+.×V
51 25 56
```

**Maximum**

Comparison

Sum-reduction of V

Product-reduction

Sum over first coordinate of M (down columns)

Sum over second coordinate of M (over rows)

Sum over last coordinate

Maximum over last coordinate

Sin squared plus Cos squared

Sin Cos X

(1-(*COS X*)*2)* 5

An identity

Ordinary matrix (+ × inner) product

An inner product

+ × inner product with vector right argument

```
      V
2 1 7
      V∘.×⍳5
 2  4  6  8 10
 1  2  3  4  5
 7 14 21 28 35
      V∘.≤⍳9
0 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1 1
      V∘.×M
 14  18   8
 10  16   2
  2  10  14

  7   9   4
  5   8   1
  1   5   7

 49  63  28
 35  56   7
  7  35  49
```

Outer product (times)

Outer product

An outer product of rank 3

A blank line between planes

MIXED FUNCTIONS

```
      Q←?10⍴5
      Q
1 4 3 4 5 4 2 1 4 2
      +/[1]Q∘.=⍳5
2 2 1 4 1
      2 1⍉M
 7  5  1
 9  8  5
 4  1  7
      ⍉M
 7  5  1
 9  8  5
 4  1  7
```

A random 10 element vector (range 1 to 5)

Ith element of result is number of occurences of the value I in Q

Ordinary transpose of M

Ordinary transpose of M (monadic)

```
      T←2 3 4⍴⍳24
      T

 1  2  3  4
 5  6  7  8
 9 10 11 12

13 14 15 16
17 18 19 20
21 22 23 24

      3 1 2⍉T

 1 13
 2 14
 3 15
 4 16

 5 17
 6 18
 7 19
 8 20

 9 21
10 22
11 23
12 24
      1 1⍉M
7 8 7
      1 1 2⍉T

 1  2  3  4
17 18 19 20
      X←○(0,⍳5)÷6
      )DIGITS 4
WAS 10
      ⍉1  2  3∘.○X

 0.000E0     1.000E0     0.000E0
 5.000E¯1    8.660E¯1    5.774E¯1
 8.660E¯1    5.000E¯1    1.732E0
 1.000E0     1.744E¯16   5.734E15
 8.660E¯1   ¯5.000E¯1   ¯1.732E0
 5.000E¯1   ¯8.660E¯1   ¯5.774E¯1
```

**An array of rank 3**

**Transpose of T (dimension of result is 3 4 2)**

**Diagonal of M**

**Diagonal section in first two coordinates of T**

**Set number of output digits to 4**

**Table of sines, cosines, and tangents in intervals of 30 degrees**



```
      Q
1 4 3 4 5 4 2 1 4 2
      3⌽Q
4 5 4 2 1 4 2 1 4 3
      ⍉3⌽Q
1 4 2 1 4 3 4 5 4 2
      0 1 2⌽[1]M

 7 8 7
 5 5 4
 1 9 1
     ⍉2⌽[2]M

 9 4 7
 8 1 5
 5 7 1
     1 2 3⌽M

 9 4 7
 1 5 8
 1 5 7
      ⌽Q
2 4 1 2 4 5 4 3 4 1

      ⌽[1]M

 1 5 7
 5 8 1
 7 9 4
      ⌽M

 4 9 7
 1 8 5
 7 5 1
```

**Rotate to left by 3 places**

**Rotate to right by 3 places**

**Rotate columns by different amounts**

**Rotation of rows all by 2 to right**

**Rotation of rows**

**Reversal of Q**

**Reversal of M along first coordinate**

**Reversal along last coordinate**

```
      U←Q>4
      U
0  0  0  0  1  0  0  0  0  0
      U/Q
5
      (~U)/Q
1  4  3  4  4  2  1  4  2
      +/U/Q
5
      1 0 1/[1]M

  7  9  4
  1  5  7
     1 0 1/M

  7  4
  5  1
  1  7
      (,M>5)/,M
7  9  8  7
      V←1 0 1 0 1
      V\⍳3
1  0  2  0  3
      V\M

  7  0  9  0  4
  5  0  8  0  1
  1  0  5  0  7
      V\'ABC'
4 5 6
      10⊥1 7 7 6
1776
      8⊥1 7 7 6
2
      (4⍴10)⊤1776
1  7  7  6
      (3⍴10)⊤1776

      1  1 7 7 6

      1 7 7 6

      ⍉⊤     ⍳2⊥1 3  5
3 8 5
      4 2  2⊥⊤3805
1  3  5
      2⊥⊤11    1 1 0
```

Compression of Q by logical vector U

Compression by not U

Compression along first coordinate of M

Compression along last coordinate

,M is 7 9 4 5 8 1 1 5 7
All elements of M which exceed 5

Expansion of iota 3

Expansion of rows of M

Expansion of literal vector inserts spaces
Base 10 value of vector 1 7 7 6

Base 8 value of 1 7 7 6

4 digit base 10 representation of number 1776
3 digit base 10 representation of 1776

Mixed base value of 1 3 25 (time radix)
Representation of number 3805 in time radix
Base 2 value

```
      M
  7  9  4
  5  8  1
  1  5  7
      )ORIGIN 0
WAS 1
      M[2;0]
1
      (,M)[(⍴M)⊥2,0]
1
      )ORIGIN 1
WAS 0
      P
2  3  5  7  11  13
      P⍳7
4
      P⍳6
7
      P,4 5 6 7
7  11  7  4
      Q←5 1 3 2 4
      R←Q⍳⍳5
      R
2  4  3  5  1
      Q[R]
1  2  3  4  5
      A←'ABCDEFGHIJKLMNOPQ'
      A←A,'RSTUVWXYZ'
      A
ABCDEFGHIJKLMNOPQRSTUVWXYZ
      A⍳'C'
3
      J←A⍳'CAT'
      J
3  1  20
A[J]
CAT
```

Indexing of matrix in 0-origin. Note relation to indexing of ravel of M

Restore 1-origin

Index of 7 in vector P
7 is 4th element of P
6 does not occur in P, hence result is 1+⍴P

A permutation vector
R is the permutation inverse to Q

A is the alphabet

Rank of letter C in alphabet is 3

```
      M←3 5⍴'THREESHORTWORDS'          A matrix of characters
      M
THREE
SHORT
WORDS
      J←A⍋M                            Ranking of M produces a matrix
      J
 20   8  18   5   5
 19   8  15  18  20
 23  15  18   4  19
      A[J]                             Indexing by a matrix produces
                                            a matrix
THREE
SHORT
WORDS
      3?5                              Random choice of 3 out of 5
5  1  2                                    without replacement
      6?5
DOMAIN ERROR
      6?5
       ^
      X←8?8                            A random permutation vector
      X
4  6  7  2  5  1  8  3
      ⍋X                               Grading of X
5  4  8  1  5  2  3  7
      X[⍋X]                            Arrange in ascending order
1  2  3  4  5  6  7  8
      X[⍒X]                            Arrange in descending order
8  7  6  5  4  3  2  1
      U←A∈'NOW IS THE TIME'            Membership
      '01'[1+U]
0000100110001110001100100 0
      U/A
EHIMNOSTW
      (⍳8)∈3 7 5
0  0  1  0  1  0  1  0
```





## ADVANCED EXAMPLES

This section presents a set of examples less elementary than those of Appendix A. These examples are all contained in Workspace *ADVANCEDEX* of Library 1. A user may therefore load, use, and trace any of the functions as an aid to understanding their behavior. Displays of intermediate results may also be inserted. For example, the statement

```
P←(P,0)+0,P
```

occurring in a function could be changed (perhaps by the use of line editing) to the following form:

```
⎕←P←(⎕←P,0)+⎕←0,P
```

Each execution of the statement will now perform as before, except that each of the results `0,P` and `P,0` and `P` will be typed out as well (in that sequence).

Programming techniques can be learned from a similar study of any well-written set of functions. All of the workspaces of library 1 may be used as a source of functions for such study.

The index origin in the workspace *ADVANCEDEX* is set to 1.

```
      )LOAD 1 ADVANCEDEX
ADVANCEDEX SAVED 07/20/68   28.12.10
      )FNS
AH      ASSOC   BIN     COMB    DTH     ENTER   F       FC
GC      GCD     GCV     HILB    HTD     IN      INV     INVP
IN1     LFC     LOOKUP  PACK    PALL    PER     PERM    PO
POL     POLY    POLYB   RESET   TIME    TRUTH   UNPACK  ZERO
      )VARS
DAH     DASSOC  DBIN    DCOMB   DDTH    DENTER  DESCRIBE
DF      DFC     DGC     DGCD    DGCV    DHILB   DHTD    DIN
DINV    DINVP   DIN1    DLFC    DLOOKUP DPACK   DPALL   DPER
DPERM   DPO     DPOL    DPOLY   DPOLYB  DTIME   DTRUTH  DUNPACK
DZERO   TIMER

      DESCRIBE

EACH OF THE VARIABLES OF THIS WORKSPACE WHICH BEGINS WITH THE
LETTER D IS THE DESCRIPTION OF THE FUNCTION WHOSE NAME IS
OBTAINED BY REMOVING THE D. FOR FURTHER DETAILS SEE APPENDIX
B OF THE APL\360 USER'S MANUAL
```

```
      DPACK
     THE FUNCTIONS PACK AND UNPACK ILLUSTRATE THE USE OF THE
ENCODE AND DECODE FUNCTIONS IN TRANSFORMING BETWEEN A FOUR-
NUMBER ENCODING OF SERIAL NUMBER (1 TO 9999), MONTH, DAY,
AND YEAR, AND A SINGLE-NUMBER ENCODING OF THE SAME DATA.

      ∇PACK[⎕]∇
    ∇ Z←PACK X
[1]   Z← 10000 12 31 100 ⊥X-1
    ∇

      ∇UNPACK[⎕]∇
    ∇ Z←UNPACK X
[1]   Z←1+ 10000 12 31 100 ⊤X
    ∇

      P←PACK 2314 7 17 68
      P
86063867
      UNPACK P
2314  7  17  68
      UNPACK PACK 2311 9 21 72
2311  9  21  72
      PACK UNPACK 92137142
92137142
      PACK 1 1 31 1
3000
      UNPACK 3000
1  1  31  1
```



```
      DENTER

THE FUNCTIONS ENTER, LOOKUP, AND RESET ILLUSTRATE A METHOD
OF CONSTRUCTING AND USING LISTS OF VARIABLE LENGTH DATA,
REPRESENTING EACH LIST BY A VECTOR OF CHARACTERS AND A
VECTOR OF INDICES. ENTER AND LOOKUP EACH REQUEST INPUT (BY ⎕)
UNTIL AN EMPTY VECTOR (CARRIAGE RETURN ALONE) IS ENTERED


RESET   RESETS LISTS (USE BEFORE ENTER AND LOOKUP)
ENTER   ACCEPTS SUCCESSIVE ITEMS OF NAMES AND DATA.
LOOKUP  PRINTS DATA ASSOCIATED WITH EACH NAME ENTERED

      ∇ENTER[⎕]∇
    ∇ ENTER;X
[1]   'ENTER NAME'
[2]   X←,⎕
[3]   →0×⍳0=⍴X
[4]   NAMES←NAMES,X
[5]   P1←P1,⍴NAMES
[6]   'ENTER DATA'
[7]   DATA←DATA,⎕
[8]   P2←P2,⍴DATA
[9]   ''
[10]  →1
    ∇

      ∇LOOKUP[⎕]∇
    ∇ LOOKUP;X;J
[1]   '?'
[2]   X←,⎕
[3]   →0×⍳0=⍴X
[4]   J←(((1↓P1)-¯1↓P1)=⍴X)/⍳¯1+⍴P1
[5]   J←(NAMES[P1[J]∘.+⍳⍴X]∧.=X)/J
[6]   →(0 1 =⍴J)/ 10 8
[7]   →1,⍴⎕←'MORE THAN ONE SUCH NAME'
[8]   DATA[P2[J]+⍳-/P2[1 0 +J]]
[9]   →1
[10]  'NO SUCH NAME'
[11]  →1
    ∇

      ∇RESET[⎕]∇
    ∇ RESET
[1]   NAMES←DATA←⍴P1←P2←0
    ∇
      RESET
      ENTER
ENTER NAME
J. ARMSTRONG
ENTER DATA
PRESIDENT

ENTER NAME
H. LEVINE
ENTER DATA
VICE-PRESIDENT

ENTER NAME

      LOOKUP
?
H. LEVINE
VICE-PRESIDENT
?
L. YAVNER
NO SUCH NAME
?
```

*DIN*

*THE FUNCTIONS IN AND IN1 TAKE TWO ARGUMENTS; THE FIRST IS A WORD (I.E., A VECTOR) WHOSE OCCURRENCES IN THE SECOND ARGUMENT ARE TO BE DETERMINED. THE RESULT IS A VECTOR OF INDICES OF THE FIRST LETTER OF EACH OCCURRENCE. THE FUNCTION IN DETERMINES ALL OCCURRENCES, WHEREAS IN1 DETERMINES ONLY ALL NON-OVERLAPPING OCCURRENCES BY FIRST APPLYING THE FUNCTION IN AND THEN SUPPRESSING ALL OVERLAPS*

```
      ∇IN[□]∇
     ∇ Z←A IN B;J
[1]    J←(A[1]=B)/⍳⍴B
[2]    J←(J≤1+(⍴B)-⍴A)/J
[3]    Z←(B[J∘.+¯1+⍳⍴A]∧.=A)/J
     ∇
      ∇IN1[□]∇
     ∇ T←A IN1 B
[1]    T←A IN B
[2]    →2×J<⍴T←(~(⍳⍴T)∈J←1+((⍴A)>|-/[1](2,1+⍴T)⍴T)⍳1)/T
     ∇

      W←'THE'
      T←'THE MEN THEN WENT HOME.'
      W IN T
1  9
      W IN1 T
1  9
      'ABA' IN 'NOWABABABABABABABA'
4  6  8  10  12  14  16
      'ABA' IN1 'NOWABABABABABABABA'
4  8  12  16
```

*DTRUTH*

*THE FUNCTION TRUTH PRODUCES THE MATRIX OF ARGUMENTS OF THE TRUTH TABLE FOR N LOGICAL VARIABLES.*

```
      ∇TRUTH[□]∇
     ∇ Z←TRUTH N
[1]    Z←2|⌊(¯1+⍳2*N)∘.÷2*N-⍳N
     ∇
      TRUTH 3
  0  0  0
  0  0  1
  0  1  0
  0  1  1
  1  0  0
  1  0  1
  1  1  0
  1  1  1
      (TRUTH 3)+.×⌽2*¯1+⍳3
0  1  2  3  4  5  6  7
```



*DGCD*

*THE FUNCTIONS GCD AND GC EACH EMPLOY THE EUCLIDEAN ALGORITHM TO PRODUCE THE GREATEST COMMON DIVISOR. GCD EMPLOYS TWO SCALAR ARGUMENTS, WHEREAS GC EMPLOYS A SINGLE ARGUMENT WHICH IS EXPECTED TO BE A TWO-ELEMENT VECTOR.*

*THE FUNCTION GCV YIELDS THE GREATEST COMMON DIVISOR OF ALL ELEMENTS OF A VECTOR OF TWO OR MORE ELEMENTS.*

```
      ∇GCD[□]∇
     ∇ Z←M GCD N
[1]    Z←M
[2]    M←M|N
[3]    N←Z
[4]    →0≠M
     ∇
      ∇GC[□]∇
     ∇ Z←GC M
[1]    →0≠1↓M←⌽M[1],Z←|/M
     ∇
      ∇GCV[□]∇
     ∇ Z←GCV W;A
[1]    →1≠⍴W←Z,(A≠0)/A←(Z←⌊/W)|W
     ∇

      84 GCD 90
6
      90 GCD 84
6
      GC 90 84
6
      GCV 90 84
6
      GCV 90 84 105
3
```

*DBIN*

*THE FUNCTION BIN PRODUCES ALL BINOMIAL COEFFICIENTS UP TO ORDER N*

```
      ∇BIN[□]∇
     ∇ Z←BIN N
[1]    Z←⌊!(0,⍳N)∘.!0,⍳N
     ∇

      BIN 4
  1  0  0  0  0
  1  1  0  0  0
  1  2  1  0  0
  1  3  3  1  0
  1  4  6  4  1
```

## DPOLY

*THE FUNCTIONS POLY, POL, PO, AND POLYB EACH EVALUATE A POLYNOMIAL (OR POLYNOMIALS), WHOSE COEFFICIENTS ARE DETERMINED BY THE FIRST ARGUMENT, AND WHOSE POINT (OR POINTS) OF EVALUATION IS DETERMINED BY THE SECOND ARGUMENT. THE COEFFICIENTS ARE IN ASCENDING ORDER OF ASSOCIATED POWERS.*

*POLY SCALAR RIGHT ARGUMENT ONLY.*
*POL SCALAR RIGHT ARGUMENT ONLY (USES INNER PRODUCT).*
*POLYB SCALAR RIGHT ARGUMENT ONLY (USES BASE VALUE).*
*PO APPLIES TO ARGUMENTS OF ANY RANK. THE VECTORS ALONG THE FIRST COORDINATE OF THE FIRST ARGUMENT ARE THE COEFFICIENTS OF THE POLYNOMIALS WHICH ARE EVALUATED FOR EACH ELEMENT OF THE SECOND ARGUMENT.*

```
      ∇POLY[□]∇
     ∇ Z←C POLY X
[1]    Z←+/C×X*¯1+⍳⍴,C
     ∇

      ∇POL[□]∇
     ∇ Z←C POL X
[1]    Z←(X*¯1+⍳⍴,C)+.×C
     ∇

      ∇POLYB[□]∇
     ∇ Z←C POLYB X
[1]    Z←X⊥¯1↓C
     ∇

      ∇PO[□]∇
     ∇ Z←C PO X
[1]    Z←(X∘.*¯1+⍳1↑⍴C)+.×C
     ∇

      C←1 2 3 4
      C POLYB 3
142
      (C POLY 3)∧.=(C POLYB 3),(C POL 3),C PO 3
1
      C PO 1 2 3 4 5 6
10 49 142 313 586 985
      □←M←⌹BIN 5
1 1 1 1 1 1
0 1 2 3 4 5
0 0 1 3 6 10
0 0 0 1 4 10
0 0 0 0 1 5
0 0 0 0 0 1
      ⌊M PO ⍳6
1 2 4 8 16 32
1 3 9 27 81 243
1 4 16 64 256 1024
1 5 25 125 625 3125
1 6 36 216 1296 7776
1 7 49 343 2401 16807
```

## DTIME

*THE FUNCTION TIME YIELDS THE AMOUNT (IN MINUTES, SECONDS, AND 60THS OF A SECOND) OF CPU TIME USED SINCE ITS LAST PREVIOUS EXECUTION. IT IS USEFUL IN MEASURING THE EXECUTION TIMES OF OTHER FUNCTIONS. THE VARIABLE 'TIMER' IS ASSIGNED THE VALUE OF THE CUMULATIVE CPU TIME AT EACH EXECUTION OF THE FUNCTION TIME.*

```
      ∇TIME[□]∇
     ∇ Z←TIME;T
[1]    Z← 60 60 60 ⊤(T←⌊21)-TIMER
[2]    TIMER←T
     ∇
```

## DCOMB

*THE FUNCTION COMB EMPLOYS RECURSIVE DEFINITION TO PRODUCE A 2!N BY 2 MATRIX OF ALL POSSIBLE PAIRS OF ELEMENTS FROM ⍳N*

*THE FUNCTION FC SHOWS AN ALTERNATE METHOD WHICH YIELDS THE SAME PAIRS BUT IN A DIFFERENT ORDER.*

*THE FUNCTION LFC EMPLOYS FC TO GENERATE LETTER PAIRS.*

```
      ∇COMB[□]∇
     ∇ C←COMB N;A;B
[1]    →0×⍳N<2
[2]    →0×⍳N=2×1↑⍴C← 1 2 ⍴ 1 2
[3]    A←COMB N-1
[4]    C←((⍴A)+(N-1),0)⍴(,A),,(⍳N-1)∘⊤0,N
     ∇
      ∇FC[□]∇
     ∇ C←FC N;A;B
[1]    B←(⍳N)∘.+N⍴0
[2]    A←(⍳N)∘.+⍳N
[3]    C←(2,N×N)⍴(,B),,A
[4]    C←⍉(C[2;]≤N)/C
     ∇
      ∇LFC[□]∇
     ∇ Z←LFC N
[1]    Z←'ABCDEFGHIJKLMNOPQRSTUVWXYZ'[FC N]
     ∇
      TIME
0 0 35
      TIME
0 0 2
      COMB 4
1 2
1 3
2 3
1 4
2 4
3 4
      TIME
0 0 12
      FC 4
1 2
1 3
1 4
2 3
2 4
3 4
      TIME
0 0 8
      LFC 4
AB
AC
AD
BC
BD
CD
      TIME
0 0 7
```

```
      Z←COMB 15
      ρZ
105  2
      TIME
0  1  4
```

```
      Z←FC 15
      ρZ
105  2
      TIME
0  0  29
```

      DDTH

*THE FUNCTIONS DTH, HTD, AND AH CONCERN HEXADECIMAL NUMBERS LIMITED TO 8 DIGITS AND EMPLOYING THE CHARACTERS 0123456789ABCDEF. NEGATIVE NUMBERS ARE REPRESENTED IN 2'S COMPLEMENT FORM, WITH ANY OF THE CHARACTERS 8 THROUGH F IN THE LEFTMOST POSITION (OF EIGHT). LEADING ZEROS MAY BE OMITTED.*

*DTH CONVERTS DECIMAL TO HEXADECIMAL*
*HTD CONVERTS HEXADECIMAL TO DECIMAL.*
*AH ADDS HEXADECIMAL NUMBERS.*

```
      ∇DTH[□]∇
    ∇ R←DTH X
[1]   R←,('0123456789ABCDEF')[1+(8ρ16)⊤X]
    ∇
      ∇HTD[□]∇
    ∇ R←HTD X
[1]   R←((8-ρ,X)ρ'0'),X
[2]   R←⌊(16⊥¯1+'0123456789ABCDEF'⍳R)-(2*32)×R[1]∈'89ABCDEF'
[3]   →4×~∧/X∈'0123456789ABCDEF'
[4]   R←''
[5]   'NUMBER IS NOT HEX'
    ∇
      ∇AH[□]∇
    ∇ R←A AH B
[1]   R←DTH(HTD A)+HTD B
    ∇
      Z←DTH 1776
      Z
000006F0
      HTD Z
1776
      Z AH Z
00000DE0
      HTD Z AH Z
3552
      HTD '000006F0'
1776
      HTD '90000000'
¯1879048192
      HTD '00049HFG'
NUMBER IS NOT HEX
```

      DZERO

*THE FUNCTION ZERO EMPLOYS THE METHOD OF FALSE POSITION TO DETERMINE TO WITHIN A TOLERANCE TOL A ROOT OF THE FUNCTION F LYING BETWEEN THE BOUNDS B[1] AND B[2]. IT IS ASSUMED THAT F B[1] AND F B[2] ARE OF OPPOSITE SIGN. THE FUNCTION F IS A SPECIFIC POLYNOMIAL, BUT CAN BE CHANGED TO ANY DESIRED FUNCTION.*

```
      ∇ZERO[□]∇
    ∇ Z←TOL ZERO B;T
[1]   →0×⍳TOL≥|T←F Z←0.5×+/B
[2]   →1,B[2⊥(0<T)≠0<F B]←Z
    ∇
      ∇F[□]∇
    ∇ Z←F X
[1]   Z← ¯20 18 3 ¯5 1 PO X
    ∇
      □←X←¯4+⍳9
¯3  ¯2  ¯1  0  1  2  3  4  5
      F X
169  12  ¯29  ¯20  ¯3  4  7  36  145
      TIME
0  1  19
      □←R←1E¯6 ZERO ¯2 ¯1
¯1 845121413
      TIME
0  2  36
      F R
7 14140814E¯7
      TIME
0  0  2
      □←F□←R←1E¯10 ZERO 1 2
1.26397094
¯1 813305062E¯11
      TIME
0  3  46
      □←F□←R←1E¯6 ZERO 1 2
1 263970852
¯8.51888359E¯7
      TIME
0  2  13
```

```
      DHILB
```

*THE FUNCTION HILB PRODUCES A HILBERT MATRIX OF ORDER N.*

```
      ∇HILB[□]∇
     ∇ Z←HILB N
[1]    Z←÷¯1+(⍳N)∘.+⍳N
     ∇
      HILB 3
```

```
1                 0.5               0.3333333333
0.5               0.3333333333      0.25
0.3333333333      0.25              0.2
```

```
      DINV
```

*THE FUNCTIONS INV AND INVP EACH PRODUCE THE INVERSE OF THE MATRIX ARGUMENT SUPPLIED, EMPLOYING GAUSS-JORDAN (I E , COMPLETE) ELIMINATION. INVP EMPLOYS PIVOTING AND INV DOES NOT*

*THE FIRST LINE APPENDS THE UNIT VECTOR 1≥⍳N AS THE LAST COLUMN OF THE ARGUMENT AND THE SECOND LINE (LINE 4 IN INVP) PERFORMS AT EACH ITERATION ONE OF THE N COMPLETE INVERSIONS REQUIRED. SEE EXERCISE 1.40 OF IVERSON, A PROGRAMMING LANGUAGE, WILEY, 1962.*

```
      ∇INV[□]∇
     ∇ Z←INV M;I;J
[1]    M←⍉(1 0 +⍴M)⍴(,⍉M),~J←1<⍳I←1↑⍴M
[2]    M←1⌽(J,1)⌽[1]M-(J×M[;1])∘.×M[1;]←M[1;]÷M[1;1]
[3]    →2×⍳0≠I←I-1
[4]    Z←M[;⍳1↑⍴M]
     ∇
      ∇INVP[□]∇
     ∇ Z←INVP M;I;J;K;P
[1]    M←⍉(1 0 +⍴M)⍴(,⍉M),~J←1<P←⍳I←1↑⍴M
[2]    M[K,1;⍳↑⍴P]←M[1,K←(|M[⍳I;1])⍳⌈/|M[⍳I;1];⍳↑⍴P]
[3]    P←1⌽P,0⍴P[K,1]←P[1,K]
[4]    M←1⌽(J,1)⌽[1]M-(J×M[;1])∘.×M[1;]←M[1;]÷M[1;1]
[5]    →2×⍳0≠I←I-1
[6]    Z←M[;⍳↑P]
     ∇
      □←N←INV M←HILB 3
```

```
 9              ¯36             30
¯36             192            ¯180
 30            ¯180             180
```

```
      M+.×N
```

```
1.000000000E0     2.842170943E¯14  ¯6.039613254E¯14
1.421085472E¯14   1.000000000E0    ¯1.065814104E¯14
4.662936703E¯15   3.197442311E¯14   1.000000000E0
```

```
      DPALL
```

*THE FUNCTION PALL PRODUCES THE MATRIX OF ALL PERMUTATIONS OF ORDER N. THE FUNCTION PERM WHICH IT USES PRODUCES THE B-TH PERMUTATION OF ORDER N BY A METHOD DUE TO L J WOODRUM*

*THE FUNCTION PER EMPLOYS RECURSIVE DEFINITION , AND PRODUCES ALL PERMUTATIONS BY A METHOD MUCH FASTER THAN THAT USED IN THE FUNCTION PALL. THE PERMUTATIONS ARE PRODUCED IN THE OPPOSITE ORDER*

```
      ∇PALL[□]∇
     ∇ Z←PALL N;I
[1]    Z←((!N),N)⍴0
[2]    I←1
[3]    Z[I;]←N PERM I
[4]    →3×(!N)≥I←I+1
     ∇
      ∇PERM[□]∇
     ∇ Z←A PERM B;I;Y
[1]    I←⍴Z←1+(⍳A)⍳B-1
[2]    →0×⍳0=I←I-1
[3]    Z[Y]←Z[Y]+Z[I]≤Z[Y←I+⍳A-I]
[4]    →2
     ∇
      ∇PER[□]∇
     ∇ P←PER M;X;Y;Z
[1]    →0×⍳M=P←1 1⍴1
[2]    Z←PER M-1
[3]    P←⍳X←0
[4]    →0×⍳M<X←X+1
[5]    Y←(~(⍳M)∈X)\Z
[6]    Y[;X]←M
[7]    P←((X×!M-1),M)⍴(,P),,Y
[8]    →4
     ∇
      PALL 3
```

```
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

```
      TIME
0 3 7
      Z←PALL 3
      TIME
0 0 49
      Z←PALL 5
      TIME
0 25 10
      Z←PER 5
      TIME
0 1 12
```

```
      DASSOC
```

THE FUNCTION ASSOC TESTS ANY PUTATIVE GROUP MULTIPLICATION TABLE M (ASSUMING GROUP ELEMENTS ⍳1⍴⍴M) FOR ASSOCIATIVITY AND YIELDS A VALUE 1 IF IT IS ASSOCIATIVE, 0 OTHERWISE

```
      ∇ASSOC[□]∇
    ∇ Z←ASSOC M
[1]   Z←∧/,M[M;]=M[;M]
    ∇

      M←(⍳5)⌽5 5⍴⍳5
      M

  2 3 4 5 1
  3 4 5 1 2
  4 5 1 2 3
  5 1 2 3 4
  1 2 3 4 5
      TIME
0 0 13
      ASSOC M
1
      TIME
0 0 9
      M←0 0 1 0 0⌽M
      M

  2 3 4 5 1
  3 4 5 1 2
  5 1 2 3 4
  5 1 2 3 4
  1 2 3 4 5
      ASSOC M
0
      TIME
0 0 10
      M←?10 10⍴10
      ⍴M
10 10
      TIME
      ⍳
      ASSOC M

      ∇F
```

```
      )LOAD 1 NEWS
SAVED  10.44.08 07/12/68
```

```
      DESCRIBE
```

THIS WORKSPACE PROVIDES INFORMATION ABOUT THE OPERATION AND USE OF APL. THE FUNCTIONS OF INTEREST TO THE USER ARE APLNOW, INDEX, PRINT, AND SCHEDULE.

APLNOW TAKES AS ITS SINGLE ARGUMENT A THREE-ELEMENT VECTOR REPRESENTING A DATE, AS MONTH, DAY, YEAR. APLNOW PRINTS NOTES ON THE STATUS OF THE APL SYSTEM; FOR INSTANCE, RECENTLY ADDED FEATURES, TEMPORARY RESTRICTIONS, OR ADVICE ON PROGRAMMING OR TERMINAL OPERATION. ONLY THOSE NOTES ENTERED INTO APLNOW ON OR AFTER THE DATE GIVEN AS AN ARGUMENT ARE PRINTED.

INDEX TAKES NO ARGUMENT. IT PRINTS INDICES, DATES, AND THE FIRST FEW WORDS OF EACH NOTE IN APLNOW.

PRINT TAKES AS ITS SINGLE ARGUMENT THE INDEX (AS INDICATED BY THE INDEX FUNCTION) OF A NOTE FROM APLNOW, AND PRINTS THE NOTE.

SCHEDULE TAKES NO ARGUMENT. IT INDICATES THE REGULAR DAILY APL SCHEDULE, AND ALL ANTICIPATED DEVIATIONS FROM THE NORMAL SCHEDULE

## BIBLIOGRAPHY


Berry, P.C., APL\360 Primer, IBM Corporation, 1968.

Berry, P.C., APL\1130 Primer, IBM Corporation, 1968.

Breed, L.M., and R.H. Lathwell, "The Implementation of APL\360", ACM Symposium on Experimental Systems for Applied Mathematics, Academic Press, 1968.

Falkoff, A.D., and K.E. Iverson,"The APL\360 Terminal System", ACM Symposium on Experimental Systems for Applied Mathematics, Academic Press, 1968.

Falkoff, A.D., K.E. Iverson, and E.H. Sussenguth, "A Formal Description of System/360", IBM Systems Journal, Volume 3, Number 3, 1964.

Iverson, K.E., A Programming Language, Wiley, 1962.

Iverson, K.E., Elementary Functions: an algorithmic treatment, Science Research Associates, 1966.

Iverson, K.E., "The Role of Computers in Teaching", Queen's Papers in Pure and Applied Mathematics, Volume 13, Queen's University, Kingston, Canada, 1968.

Lathwell, R.H., APL\360: Operator's Manual, IBM Corporation, 1968.

Lathwell, R.H., APL\360: System Generation and Library Maintenance, IBM Corporation, 1968.

Pakin, S., APL\360 Reference Manual, Science Research Associates, 1967.

Rose, A.J., Videotaped APL Course, IBM Corporation, 1968.

Smillie, K.W., Statpack 1: An APL Statistical Package, Publication No. 9, Department of Computing Science, University of Alberta, Edmonton, Canada, 1968.


Volume II

# APL\360: Operator's Manual


R. H. Lathwell



© International Business Machines Corporation, 1968

ACKNOWLEDGEMENTS

The basic operator procedures were developed by the system designers listed in the companion User's Manual. A. D. Falkoff contributed significantly to the organization of the present manual, which also benefitted from critical reading by J. G. Arnold*, K. E. Iverson, and E. E. McDonnell.

A special note of gratitude is due to Mrs. G. E. Barrett, operator extraordinary, for her patience, sense of humor, and helpful suggestions during the early trying months of operation.

* Industry Development, IBM Corporation, White Plains, N.Y.





TABLE OF CONTENTS

INTRODUCTION 152
DOS OPERATION 152
Messages from the system, Job step CPU accounting, Multiprogramming, Operator commands
INITIALIZING APL\360 157
THE APL\360 RECORDING TERMINAL 159
APL\360 PRIVILEGED COMMANDS 160
Privileged commands, definitions, Trouble Reports
APL\360 SYSTEM COMMANDS 164
APL\360 OPERATOR'S LOG 166
TERMINATING APL\360 169

APPENDIX A 170
SYSLOG messages during APL initialization

LIST OF ILLUSTRATIONS

Table 1 RECOMMENDED FEATURES AND OPTIONS FOR TERMINALS 154

Table 2 NOTES AND RESTRICTIONS ON DOS OPERATING COMMANDS 156

Table 3 APL\360 SYSTEM COMMANDS 165

Table 4 SENSE BYTE INFORMATION FOR 2311 AND 2314 DISKS 168

## INTRODUCTION

APL\360 consists of a modified DOS Version 3 plus the APL\360 Supervisor and Interpreter. The modified DOS appears essentially the same as the standard DOS release of the corresponding change level.

DOS operating procedure is fully described in C22-5022, DOS Operating Guide, and familiarity with that manual is assumed. The use of APL\360 is fully described in APL\360: User's Manual, and familiarity with parts 1 and 2 of that publication is also assumed.

If the modified DOS is used without APL, it is operated as described in C22-5022, DOS Operating Guide. When APL is running, an additional console is required for recording APL system information. The recommended device for this console is an IBM 2741 Communications Terminal with the interrupt feature. See Table 1 for details of communication terminals.

## DOS OPERATION

Messages from the system. DOS system messages have not been changed. S and W type messages have been augmented to display a distinctive pattern in the Instruction Address Register if a failure occurs which causes the system to enter the WAIT state with all interrupts disabled.

When an S condition (CPU, Storage, or Channel failure) occurs, the modified DOS Supervisor attempts an automatic recovery. If the recovery appears to be successful, a message of the form

id PROB PROG MC

is printed on SYSLOG. id identifies the partition which was active when the machine check occurred. If the recovery attempt is unsuccessful, DOS enters the WAIT state with all interrupts disabled and with the pattern

X'7FFFF7'

displayed in the IAR.

When the system stops, the SEREP should be run, and the resulting printout given to the IBM Customer Engineer. The IPL procedure is required to restart the system.

There are three types of W conditions, distinguishable by the pattern displayed in the IAR as well as by information placed in low storage

X'70F0F0'

A program check occurred within the DOS or APL supervisor. (This is a relatively rare occurrence.) A storage dump should be obtained and forwarded with a copy of the APL Linkage Edit map and a DOS Supervisor listing to the APL program author.

X'555555'

An I/O error occurred on SYSRES which prevented a phase from being loaded into storage. SYSRES not in a ready condition is one of the possible causes. If this condition persists when SYSRES otherwise appears to be operational, regard it as a channel or control unit failure.

X'505050'

A catastrophic channel or control unit failure occurred which was detected in channel status but did not result in a machine check interruption. When this happens frequently, the IBM Customer Engineer should be notified. The failing channel and unit address are placed in low storage as described in C22-5022, DOS Operating Guide.

Note: On System/360 Models 65 and larger, the value displayed in the IAR may be 8 greater than shown above

**Job step CPU accounting.** Partition Job Step CPU time accounting has been added to DOS. Whenever a job step terminates, a message of the form:

id time jobname CPU cputime

is printed on SYSLOG. The time printed to the left is the time of day at which the job step terminated. The CPU accounting printed at the completion of the IPL procedure is meaningless.

**Multiprogramming.** Multiprogramming commands are as described in the DOS Operating Guide. The following restriction should be noted.

AN ALLOC COMMAND GIVEN WHILE APL IS ACTIVE WILL RESULT IN APL TERMINATION ACCOMPANIED BY A MESSAGE OF THE FOLLOWING FORM:

F1 JOB APLSMPS CANCELLED DUE TO PROGRAM CHECK.

| FEATURE OR OPTION | 1050 | 2740-1 | 2741 |
|---|---|---|---|
| Control Unit | 1051-2 | | |
| Voltage (115 AC), Non-lock plug | 9881 | 9881 | 9881 |
| Dataset Attachment | 9114 | 9114 | 9114 |
| Dial Up | NR | 3255 | 3255 |
| Transmit Control | NR | 8028 | NR |
| Automatic EOB | RPQ E27283 | Do not use | NR |
| Typamatic Keys | NA | NA | 8341 |
| Interrupt | RPQ E27428 | RPQ F17913 | 4708 |
| Text Time-out Suppression | 9698 | NR | NR |
| First Printer Attachment | 4408 | NR | NR |
| Automatic Ribbon Shift Select | 1295 | NA | NA |
| Typing Table | 9705 | NR | NR |
| Printer-Keyboard | 1052-2 | | |
| APL Printing Element, PTTC/BCD | 1167988 | 1167988 | 1167988 |
| or Standard Selectric® | NA | 1167987 | 1167987 |
| Keys, APL Keyboard | RPQ M40174 | RPQ M40174 | RPQ M40174 |
| Character Spacing, 10 per inch | 9104 | 9104 | 9104 |
| Line Feeding, 6 per inch | 9435 | 9435 | 9435 |
| Accelerated Carrier Return | 1006 | NA | NA |

Notes. NR: feature is standard equipment, or is not required.
NA: not available (July 1968).
The numbers are IBM-domestic identifications.

Table 1: RECOMMENDED FEATURES AND OPTIONS FOR TERMINALS

Operator commands. For the complete specifications of DOS operator commands, see C22-5022, DOS Operating Guide. Those commands which carry special restrictions or notes when used with APL are given in Table 2.

| COMMAND | NOTES |
|---|---|
| ALLOC id=nK | Must not be given while APL is active, since it will cause APL to terminate with a program check. |
| ASSGN SYSxxx,X'cuu' | Sharing an APL library or swapping disk with another partition will degrade system performance. |
| HOLD id | This version of DOS allows foreground units to be pre-assigned during system generation and held. |
| MSG id | APL does not accept messages from SYSLOG. |
| SET | For accounting reasons, the time and date cannot be changed while APL is running, and care should be taken to ensure that the correct values are set prior to APL initiation. While APL is active, SET CLOCK is completely ignored, and SET DATE is ignored by APL. |
| TIMER id | The timer must be assigned to the foreground 1 partition when APL is initiated, otherwise APL initialization will terminate because of an illegal SVC. |

Table 2: NOTES AND RESTRICTIONS ON DOS OPERATOR COMMANDS

## INITIATING APL\360

APL is initiated by a series of foreground initiation commands entered from SYSLOG or from a card reader. A complete procedure is shown below. Physical unit addresses and storage allocation conform to the configuration given in APL\360: System Generation and Library Maintenance, Appendix B: Example System Generation.

If an error is encountered during initialization, APL will print a message on SYSLOG and terminate. Details of these messages appear in Appendix A.

Example initiation procedures. In SYSLOG examples, messages printed by DOS are upper case and those typed by the DOS Operator are lower case.

Mount APL swapping and library disks (on 172 and 173).

SYSLOG:

```
BG 1100A READY FOR COMMUNICATIONS
BG alloc f1=210K                    see note 1
BG stop                             see note 2
(press 1052 REQUEST)
AR 1160A READY FOR COMMUNICATIONS
AR start f1                         see note 3
```

a) Initiation from SYSLOG:

```
F1 assgn sys005,x'172'              see note 4
F1 assgn sys006,x'172'
F1 assgn sys007,x'173'
F1 exec aplsmps                     see note 5
F1 APL IS RUNNING                   see note 6
```

b) Initiation from the card reader. Cards in reader:

```
ASSGN SYS005,X'172'                 see note 4
ASSGN SYS006,X'172'
ASSGN SYS007,X'173'
EXEC APLSMPS                        see note 5
```

SYSLOG:

```
F1 read x'00c'                      see note 7
F1 APL IS RUNNING
```

Notes:

1. The required storage allocation may be pre-specified. (See APL\360: System Generation and Library Maintenance, Step 4). If this has been correctly done, and the allocation has not been changed by the DOS operator, the alloc command is not required.

2. If a background batch stream has been interrupted with a pause command so that the alloc command can be given, resume background processing by entering EOB rather than stop.

3. This command requests initiation of the foreground 1 partition. APL always runs in foreground 1.

4. Foreground logical units may be pre-assigned (see APL\360: System Generation and Library Maintenance, Step 4). If the correct units for APL have been assigned, and the assignments have not been changed by the DOS operator, the assgn statements are not required.

5. The exec aplsmps statement causes APL to be loaded into storage and APL initialization to begin.

6. Indicates that APL initialization was successful. (See Appendix B for details of messages printed on SYSLOG).

7. Specifies that foreground initiation commands are to be read from card reader 00c.

After the message APL IS RUNNING, the operator should sign on the APL recording terminal. The sign-on procedure is described in detail in APL\360: User's Manual.

After connecting the recording terminal, the operator signs on by entering

```
)314159
```

When this is correctly entered, APL responds with

*OPR*) time date *OPERATOR*

APL is now fully initialized, and APL users can begin signing on.

THE APL\360 RECORDING TERMINAL

Any terminal signed on with the account number 314159, which is the operator's number, is the APL recording terminal. It is similar to any other APL terminal with the following exceptions:

1. Its keyboard is normally locked. The operator must signal attention to unlock the keyboard, prior to making an entry.

2. It is always privileged and all privileged commands and operations can be executed from it.

3. Output to it cannot be cancelled by signalling attention. Interrupted output is retransmitted.

4. It receives messages from APL.

The recording terminal must be signed on and locked to allow other APL users to sign on.

The recommended device for the APL recording terminal is an IBM 2741 Communications Terminal with the Interrupt feature. It may be connected by dial-up data set or by a fixed connection. Other terminals may be used, but some means of signalling attention is required.

See Table 1 for details of terminals recommended for APL.

APL recognizes the operator by his account number, which is 314159. The first terminal signing on with this number becomes the recording terminal: it is therefore advisable to protect the operator's number with a password.

APL\360 PRIVILEGED COMMANDS

The system commands described here are privileged, and may be used only at a privileged terminal. The recording terminal is the only terminal which automatically becomes privileged. Like all APL system commands, these must begin with a right parenthesis and end with a carriage return. Items enclosed in brackets are optional; all others must be given. The brackets are not typed when entering a command. Items are separated by spaces.

Definitions

| | |
|---|---|
| number | Account number or public library number. Numbers 1 through 999 are reserved for public libraries. User account numbers are numbers 1000 through 2147483647. A user's account number is also his library number. |
| key or lock | A password of 1 to 8 alphabetic or numeric characters, set off from the preceding text by a colon. |
| wsid | Library number and workspace name, or workspace name alone, as required. |
| userid | Identification which is printed with the sign-on acknowledgement. It consists of from 1 to 11 characters, the first of which must be alphabetic; the remainder alphabetic or numeric. |
| port | A number assigned to a Transmission Control Unit line position by APL during system generation. |
| quota | One less than the maximum number of stored workspaces a user may have at one time. |

Privileged Command Definitions. Numbered responses refer to Table 3, System Commands, and the four additional trouble reports (17 to 20) listed in a later section.

)ADD number userid [lock] quota-adjustment

Purpose: Enroll a user, change a userid or account lock, adjust a quota, or add a public library.
Normal response: none.
Trouble reports: 1, 16, 17, 18, or 19.
Notes: A password may be changed by the APL operator but may not be removed. If no lock is given, the password remains unchanged.
The quota-adjustment is additive - an integer increases it by the specified amount, an integer preceded by a negative sign (upper case 2) decreases it. A quota-adjustment of zero leaves the quota unchanged.
Although they must be given, the name, userid and quota-adjustment for a public library are meaningless. However, each public library must be added before it becomes available for use.

)BOUNCE port

Purpose: Force an ending to a user's work session.
Normal response: A sign-off message when the forced sign-off is complete.
Trouble reports: 16.
Notes: If no response is received after several seconds, retry the command. There are certain periods during execution when a user cannot be forced off.

)DELETE number

Purpose: Remove an account number or public library number from the system, or delete a public library.
Normal response: none.
Trouble reports: 1, 9, 16, or )DROP WS AND SAVE.
Notes: All workspaces associated with the library must be dropped before a )DELETE command will be accepted.

)HI up to 120 characters of text

Purpose: Deliver a message to users as they sign on.
Normal response: none.
Trouble reports: 16.
Notes: )HI followed immediately by a carriage return deletes the previous message.

)PA up to 120 characters of text.

Purpose: Broadcast a message to all signed-on users.
Normal response: none.
Trouble reports: 16.
Notes: A broadcast message will interrupt certain user operations, and so should be used only when an urgent message must be transmitted.

)HIPA up to 120 characters of text.

Purpose: Simultaneous )HI and )PA commands.
Normal response: none.
Trouble reports: 16.

)LOCK number

Purpose: Bar a user from using APL\360.
Normal response: none.
Trouble reports: 1, 16.
Notes: This does not affect the locked user's library. When the locked user attempts to sign on he will receive the trouble report *NUMBER LOCKED OUT* .

)PRIV port

Purpose: Privilege the terminal on the specified port.
Normal response: none.
Trouble reports: 16.
Notes: A privileged terminal has access to all privileged system commands and operations until signed off. The command has no effect if there is no terminal signed on the specified port. The privilege may be removed only with a )BOUNCE command.

)UNLOCK number

Purpose: Reinstate a previously barred user.
Normal response: none.
Trouble reports: 1, 16.

)PORTS

Purpose: List ports in use.
Normal response: A list of ports in use and corresponding account numbers and user ids.
Trouble reports: 16.

Trouble reports 1 through 16 are given in Table 3 and are discussed in detail in APL\360: User's Manual, parts 1 and 2.

The following additional trouble reports may be received at a privileged terminal. In all cases, the operation requested by the system command which resulted in the error is not performed.

17 *NO SPACE*

The library disks are full. Workspaces must either be deleted to provide room for new saved workspaces, or the amount of disk space available for the APL users' library must be increased.

18 *MAN TABLE FULL*

There is not enough space in the APL user directories to enroll more users. Accounts must be deleted, or the number of directories increased before new accounts can be added.

19 *LIBRARY TABLE FULL*

There is not enough space in the APL library directories to contain pointers to new saved workspaces. Workspaces must be dropped, or the number of directories increased before new workspaces can be saved.

20 *DIRECTORY ERROR*

This indicates that the library directories have been damaged during APL operation. They may have to be restored from a backup tape.

## APL \360 SYSTEM COMMANDS

These commands are shown in Table 3 and described in APL \360: User's Manual. Additional functions or restrictions at a privileged terminal are noted below.

*)OFF*
*)CONTINUE*

The APL Operator signs off in the same manner as another APL user. Users already signed on continue unaffected, but no further users may sign on.

*)LIB*
*)SAVE*
*)DROP*

A privileged terminal may perform these operations with respect to any library. A workspace stored in another user's library comes under his full control as if he had saved it himself.

*)MSG*
*)MSGN*

There is no distinction between these commands, since the keyboard of the recording terminal always remains locked after the message has been delivered. Messages from the operator are never preceded by the character *R*.

*)COPY*
*)PCOPY*

Copy operations may abort with unpredictable errors when attempted from the recording terminal.

Passwords on saved workspaces are inviolate and there is no way that even a privileged terminal can learn a password. If a user saves a locked workspace and then forgets the password, his only recourse is to drop it.

Account passwords can be changed at a privileged terminal by an *)ADD* command, but cannot be removed.

| Reference and Purpose<br>COMMAND FORM[1,2,3] | NORMAL RESPONSE | TROUBLE REPORTS[4] |
|---|---|---|
| TC1 Sign on designated user and start a work session. | | |
| )NUMBER [KEY] | [TEXT]; PORT,TIME,DATE,USER; SYSTEM; [*SAVED*,TIME,DATE] | 1 2 3 4 5 |
| TC2 End work session. | | |
| )OFF [LOCK] | PORT,TIME,DATE,USER CODE; TIME USED | 16 |
| TC3 End work session and hold dial-up connection. | | |
| )OFF HOLD [LOCK] | PORT,TIME,DATE,USER CODE; TIME USED | 16 |
| TC4 End work session and store active workspace. | | |
| )CONTINUE [LOCK] | [TIME,DATE,*CONTINUE*]; PORT,TIME,DATE,USER CODE; TIME USED | 6 16 |
| TC5 End work session, store active workspace, and hold dial-up connection. | | |
| )CONTINUE HOLD [LOCK] | [TIME,DATE,*CONTINUE*]; PORT,TIME,DATE,USER CODE; TIME USED | 6 16 |
| WC1 Activate a clear workspace. | | |
| )CLEAR | CLEAR WS | 16 |
| WC2 Activate a copy of a stored workspace. | | |
| )LOAD WSID [KEY] | *SAVED*,TIME,DATE | 7 8 16 |
| WC3 Copy a global object from a stored workspace. | | |
| )COPY WSID [KEY] NAME | *SAVED*,TIME,DATE | 6 7 8 9 10 16 |
| WC3a Copy all global objects from a stored workspace. | | |
| )COPY WSID [KEY] | *SAVED*,TIME,DATE | 6 7 8 10 16 |
| WC4 Copy a global object from a stored workspace, protecting active workspace. | | |
| )PCOPY WSID [KEY] NAME | *SAVED*,TIME,DATE; [*NOT COPIED*:,LIST OF OBJECTS] | 6 7 8 9 10 16 |
| WC4a Copy all global objects from a stored workspace, protecting active workspace. | | |
| )PCOPY WSID [KEY] | *SAVED*,TIME,DATE; [*NOT COPIED*:,LIST OF OBJECTS] | 6 7 8 10 16 |
| WC5 Gather objects into a group. | | |
| )GROUP NAME[S] | NONE | 11 16 |
| WC6 Erase global objects. | | |
| )ERASE NAME[S] | [*NOT ERASED*:,LIST OF OBJECTS] | 16 |
| WC7 Set index origin for array operations. | | |
| )ORIGIN INTEGER,0-1 | *WAS*,FORMER ORIGIN | 16 |
| WC8 Set maximum for significant digits in output. | | |
| )DIGITS INTEGER,1-16 | *WAS*,FORMER MAXIMUM | 16 |
| WC9 Set maximum width for an output line. | | |
| )WIDTH INTEGER,30-130 | *WAS*,FORMER WIDTH | 16 |
| WC10 Change workspace identification. | | |
| )WSID WSID | *WAS*,FORMER WSID | 16 |
| LC1 Re-store a copy of the active workspace. | | |
| )SAVE | TIME,DATE,WSID | 6 12 13 14 16 |
| LC1a Store a copy of the active workspace. | | |
| )SAVE WSID [LOCK] | TIME,DATE | 6 12 13 14 16 |
| LC2 Erase a stored workspace. | | |
| )DROP WSID | TIME,DATE | 7 14 16 |
| IQ1 List names of defined functions. | | |
| )FNS [LETTER] | FUNCTION NAMES | 16 |
| IQ2 List names of global variables. | | |
| )VARS [LETTER] | VARIABLE NAMES | 16 |
| IQ3 List names of groups. | | |
| )GRPS [LETTER] | GROUP NAMES | 16 |
| IQ4 List membership of designated group. | | |
| )GRP NAME | FUNCTION NAMES,VARIABLE NAMES | 16 |
| IQ5 List halted functions (state indicator). | | |
| )SI | SEQUENCE OF HALTED FUNCTIONS | 16 |
| IQ6 List halted functions and associated local variables (augmented state indicator). | | |
| )SIV | SEQUENCE OF HALTED FUNCTIONS WITH NAMES OF LOCAL VARIABLES | 16 |
| IQ7 Give identification of active workspace. | | |
| )WSID | WSID | 16 |
| IQ8 List names of workspaces in designated library. | | |
| )LIB [NUMBER] | NAMES OF STORED WORKSPACES | 14 16 |
| IQ9 List ports in use and codes of connected users. | | |
| )PORTS | PORT NUMBERS AND ASSOCIATED USER CODES | 16 |
| IQ10 List port numbers associated with designated user code. | | |
| )PORTS CODE | PORT NUMBERS | 16 |
| CM1 Address text to designated port. | | |
| )MSGN PORT [TEXT] | *SENT* | 15 16 |
| CM2 Address text to designated port, and lock keyboard. | | |
| )MSG PORT [TEXT] | *SENT* | 15 16 |
| CM3 Address text to recording terminal (APL Operator). | | |
| )OPRN [TEXT] | *SENT* | 15 16 |
| CM4 Address text to recording terminal (APL Operator), and lock keyboard. | | |
| )OPR [TEXT] | *SENT* | 15 16 |

**TROUBLE REPORT FORMS**

1 *NUMBER NOT IN SYSTEM*
2 *INCORRECT SIGN-ON*
3 *ALREADY SIGNED ON*
4 *NUMBER IN USE*
5 *NUMBER LOCKED OUT*
6 *NOT WITH OPEN DEFINITION*
7 *WS NOT FOUND*
8 *WS LOCKED*
9 *OBJECT NOT FOUND*
10 *WS FULL*
11 *NOT GROUPED, NAME IN USE*
12 *NOT SAVED, WS QUOTA USED UP*
13 *NOT SAVED, THIS WS IS* **WSID**
14 *IMPROPER LIBRARY REFERENCE*
15 *MESSAGE LOST*
16 *INCORRECT COMMAND*

Notes: 1. Items in brackets are optional.
2. KEY or LOCK: a password set off from preceding text by a colon.
3. WSID: library number and workspace name, or workspace name alone, as required.
4. See insert table of trouble report forms.

Table 3: APL\360 SYSTEM COMMANDS

APL\360 OPERATOR'S LOG

There are three types of messages logged on the recording terminal:

1. Messages in response to commands or actions by the APL Operator. These are described in APL\360: User's Manual, and in the section of this manual dealing with APL\360 privileged commands.

2. Messages from APL users. APL users send messages to the operator with either the )OPR or the )OPRN command. These messages have the form:

```
317:R WHEN IS THE SYSTEM GOING DOWN?
062: PLEASE SEND UP SOME COFFEE.
```

The number preceding the colon is the number of the port from which the message was sent. If the character R is printed, the sender has used )OPR and left his keyboard locked so that he may receive a reply. If not, the user has used )OPRN and is not awaiting a reply.

Any message to the operator from a user before he is signed on locks his keyboard, and he may not proceed with his sign-on until the operator has replied.

3. Disk error log.

```
***=ocuu,SENSE=sense,CH=track
```

APL will attempt to recover by retrying the disk I/O operation 10 times. A sense message is logged after each unsuccessful try.

The code printed for o indicates the APL disk operation in progress when the error occurred:

2 Swapping workspace read.
If unrecoverable, the swapping tracks on which the error occurred will be abandoned, and the corresponding terminal will receive the message:
```
CLEAR WS
```
If insufficient swapping area remains, APL stops as described in operation 6.

4 Directory read.
The APL Supervisor will attempt to read the alternate copy of the directory.

6 Write into the APL library.
If unrecoverable, the APL Supervisor stops by entering a program loop with all interrupts disabled. This only happens when there is something seriously wrong with the disk, and the IBM Customer Engineer should be notified.

8 Alternate directory write.
If unrecoverable, APL stops as described in operation 6.

A Read from the APL library.
If unrecoverable, the library read operation will be abandoned, and the initiating terminal will receive the message
```
CLEAR WS
```

C Second directory read.
APL will attempt to read the alternate copy.

E Primary directory write.
If unrecoverable, APL stops as described in operation 6.

cuu is the channel and unit address of the disk on which the error occurred.

sense contains sense bytes 0 through 3, indicating the type of error that occurred.

track is the hexadecimal address of the cylinder and track on which the error occurred.

Example sense message and error analysis.

```
7.,SENSE=00040340,CH=0101
```

| | |
|---|---|
| Operation: | Swapping read. |
| Device: | 172 |
| Error: | No record found. |
| Address: | Cylinder 1, track 5. |

| BYTE 0 | | BYTE 2 | |
|---|---|---|---|
| Bit | Meaning | Bit | Meaning |
| 0 | Command reject | 0 | Unsafe |
| 1 | Intervention required | 1 | Not used |
| 2 | Bus out check | 2 | SERDES check |
| 3 | Equipment check | 3 | CU tag line check |
| 4 | Data check | 4 | ALU check |
| 5 | Overrun | 5 | Unselected status |
| 6 | Track condition | 6 | Not used |
| 7 | Seek check | 7 | Not used |
| BYTE 1 | | BYTE 3 | |
| Bit | Meaning | Bit | Meaning |
| 0 | Data check in count | 0 | Ready |
| 1 | Track overrun | 1 | On line |
| 2 | End of cylinder | 2 | Unsafe read |
| 3 | Invalid sequence | 3 | Unsafe write |
| 4 | No record found | 4 | Not used |
| 5 | File protected | 5 | End of cylinder |
| 6 | Missing address marker | 6 | Not used |
| 7 | Overflow incomplete | 7 | Seek incomplete |

Table 4: SENSE BYTE INFORMATION FOR 2311 AND 2314 DISKS



## TERMINATING APL\360

The foreground 1 partition containing APL may be cancelled whenever the DOS Operator is able to communicate with the DOS Attention Routine. APL should be cancelled only after all users, including the APL Operator, have signed off.

The following sequence on SYSLOG cancels APL:

```
(press 1052 request)
AR 1160A READY FOR COMMUNICATIONS
AR cancel f1
AR (eob)
F1 time     APLSMPS  CPU cputime
F1 JOB APLSMPS CANCELLED DUE TO OPERATOR INTERVENTION
```

## *OPFNS*, THE OPERATOR'S WORKSPACE

A workspace of system functions is distributed in the APL Operators library. Before it can be properly used, this workspace must be initialized as follows:

```
     )LOAD OPFNS
SAVED  time date
     INITIALIZE
```

The initialization function will request certain information and initialize the workspace. When the initialization has been accomplished, it should be saved again as follows:

```
     )SAVE OPFNS
time date
```

A function which describes the use of the Operator's workspace can be executed by loading it and entering:

```
     OPFNS
```

The functions in the Operator's workspace may be used only from a privileged terminal.

## APPENDIX A

## SYSLOG MESSAGES DURING APL INITIALIZATION

F1 APL IS RUNNING

Reason: The APL supervisor has taken control and the APL Operator may now sign on.

F1 JOB APLSMPS CANCELLED DUE TO INVALID ADDRESS

Reason: The address specified in the PHASE APLSMPS statement included in the APL Linkage Edit (System Generation Step 9) is lower than the origin of the foreground 1 area.
Correction: Use an alloc command from SYSLOG to allocate the required foreground 1 storage.

F1 INSUFFICIENT SWAP AREA

Reason: The number of cylinders allocated for swapping is too few.
Correction: Repeat as necessary, System Generation Steps 10, 11, and 12.

F1 INSUFFICIENT CORE STORAGE

Reason: The address specified in the PHASE APLSMPS statement included in the APL Linkage Edit is too high.
Correction: Repeat System Generation Step 9.

F1 ERROR IN DS LABEL, SYSnnn

Reason: The disk mounted on the file assigned to SYSnnn does not contain the label expected by APL. Most often, the wrong disks are mounted or the APL disks are incorrectly assigned.
Correction: When this message is received during APL initiation immediately after a system generation, check and repeat as necessary, System Generation Steps 10, 11, and 12. In other cases, check APL disk mounting and assignment.

F1 LIBRARY DEVICES DIFFER

Reason: APL Library disks must all be 2311's or all 2314's.

F1 INVALID DEVICE TYPE, SYSnnn

Reason: SYSnnn is not assigned to a disk.
Correction: Assign SYSnnn to the proper disk file.

F1 OVERLAPPING LIBRARY EXTENTS

Reason: Library extents contained in two labels on the same disk overlap.
Correction: Repeat as necessary, System Generation Steps 10, 11, and 12.

F1 JOB APLSMPS CANCELLED DUE TO DEVICE NOT ASSIGNED

Correction: If this message accompanies any of the preceding three, it can be ignored. If not, use a LISTIO F1 command to determine which logical units specified in System Generation Step 7 are not assigned.

F1 INTERVAL TIMER NOT STEPPING
F1 PLEASE ENABLE AND SET CLOCK.

Reason: The Interval Timer is disabled.
Correction: Enable the timer and set the correct time of day with a set command in the background partition.

Volume III

# APL\360: System Generation and Maintenance

R. H. Lathwell

© International Business Machines Corporation, 1968

## ACKNOWLEDGEMENTS

The system generation procedures and the modifications necessary to run APL under DOS were developed by R. D. Moore* and R. H. Lathwell. The library maintenance procedures were designed by L. M. Breed. Helpful advice on DOS was provided by H. P. Cate Jr.†

The development was also furthered by the experience of those who generated and ran early versions of the system at other installations, notably W. H. Niehoff‡, T. J. Baribault**, and Professor W. S. Adams and his staff at the University of Alberta.

The organization of the manual owes much to the advice of E. M. Sharp†† and A. D. Falkoff. For critical reading of drafts, the author is indebted to J. G. Arnold‡‡, A. D. Falkoff, K. E. Iverson, and E. E. McDonnell.

* I. P. Sharp Associates, Toronto, Canada.
† Programming Systems, IBM Corporation, Endicott, N.Y.
‡ Engineering Education, IBM Corporation, Endicott, N.Y.
** Quebec Office, IBM Corporation, Quebec, Canada.
†† Advanced Systems Technology, IBM Corporation, Poughkeepsie, N.Y.
‡‡ Industry Development, IBM Corporation, White Plains, N.Y.

TABLE OF CONTENTS


INTRODUCTION 176
Minimum system configuration, Recommended terminal devices, System resources used, Modifications to DOS
THE APL 360 SYSTEM GENERATION PROCEDURE 179
Steps required to generate DOS and APL
SOME SYSTEM MAINTENANCE NOTES 1[illegible]
APL\360 LIBRARY MAINTENANCE 197
APL UTILITY FUNCTIONS 198

APPENDIX A 201
APL system generation modules

APPENDIX B 203
Example system generation

APPENDIX C 210
BPS UTILITY messages

APPENDIX D 212
APL UTILITY error messages


LIST OF ILLUSTRATIONS


Table 1 RECOMMENDED FEATURES AND OPTIONS FOR TERMINALS 177

Table 2 REQUIRED DOS SUPERVISOR PARAMETER VALUES 183

Table 3 APL CONFIGURATION MACROS 19[illegible]

Table 4 APL UTILITY CONTROL CARDS 200


This [illegible] been intentionally left blank.

| FEATURE OR OPTION | 1050 | 2740-1 | 2741 |
|---|---|---|---|
| Control Unit | 1051-2 | | |
| Voltage (115 AC), Non-lock plug | 9881 | 9881 | 9881 |
| Dataset Attachment | 9114 | 9114 | 9114 |
| Dial Up | NR | 3255 | 3255 |
| Transmit Control | NR | 8028 | NR |
| Automatic EOB | RPQ E27283 | Do not use | NR |
| Typamatic Keys | NA | NA | 8341 |
| Interrupt | RPQ E27428 | RPQ F17913 | 4708 |
| Text Time-out Suppression | 9698 | NR | NR |
| First Printer Attachment | 4408 | NR | NR |
| Automatic Ribbon Shift Select | 1295 | NA | NA |
| Typing Table | 9705 | NR | NR |
| Printer-Keyboard | 1052-2 | | |
| APL Printing Element, PTTC/BCD | 1167988 | 1167988 | 1167988 |
| or Standard Selectrico | NA | 1167987 | 1167987 |
| Keys, APL Keyboard | RPQ M40174 | RPQ M40174 | RPQ M40174 |
| Character Spacing, 10 per inch | 9104 | 9104 | 9104 |
| Line Feeding, 6 per inch | 9435 | 9435 | 9435 |
| Accelerated Carrier Return | 1006 | NA | NA |

Notes. NR: feature is standard equipment, or is not required.
NA: not available (July 1968).
The numbers are IBM-domestic identifications.

Table 1: RECOMMENDED FEATURES AND OPTIONS FOR TERMINALS

INTRODUCTION

Familiarity with the following publications is assumed:

APL\360: Operator's Manual
C24-5022 DOS Operating Guide
C24-5033 DOS System Generation and Maintenance

Minimum System Configuration

Machine Requirements:

2040 CPU.
256k bytes of main storage.
Timer feature.
Storage protection feature.
Universal instruction set.
Multiplexor channel.
One selector channel.
One card reader.
One card punch.
One printer.
One 1052 Printer keyboard.
Three 2311 Disk Storage Drives, or One
2314 Direct Access Storage Facility.
One 2702 or 2703 Transmission Control Unit.
One 2400-series 9 track magnetic tape unit.

Recommended terminal devices are described in Table 1.

Terminals may be connected either by Western Electric 103A2 (or equivalent) dial-up data-sets, by IBM Limited Distance Four Wire Modems, or by leased line modems such as WTC 3976 and Western Electric 103F2. Non dial-up terminals require an interrupt facility.

System resources used by APL\360, while active:

Approximately 200,000 bytes of main storage.
A minimum of one 2311 (or 2314) Disk Storage Drive.
A minimum of one 2702 line position.

Modifications to DOS have been confined to the DOS core resident supervisor, program number 360N-CL-453, in such a way as to maintain compatibility with DOS Version 3 if the supervisor is generated as described in this manual. The following areas are affected:

Old and new PSW areas of prefix storage.
Diagnostic scanout area.
General Cancel.
General Exit.
Communications Region.
General Entry.
Channel Scheduler.
Start I/O Routine.
I/O Interrupt Routine.
Error Recovery Exits.
Fetch Subroutine.
SVC Interrupt Routines.
Program Check Interrupt Routine.
Constants and Equates.

## THE APL\360 SYSTEM GENERATION PROCEDURE

The procedure required to generate an operational system is similar to the procedure described in C24-5033, DOS System Generation and Maintenance. All of the steps below must be completed before APL can be fully operational, and they must be performed in the order shown.

Separate maintenance and operational SYSRES volumes may be created as described in DOS System Generation and Maintenance. The source and relocatable modules listed in Appendix A may be deleted from the Relocatable and Source Statement Libraries on the operational pack only.

Steps required to generate DOS and APL.

1. Restore the distributed system tape to a system residence disk.
2. IPL the restored system.
3. Initialize the SYSRES label cylinder.
4. Assemble and catalog the DOS supervisor.
5. Assemble and catalog the APL configuration.
6. Estimate APL swapping and storage requirements.
7. Assemble and catalog APL disk parameters.
8. Linkage edit the APL utility program.
9. Linkage edit APL.
10. Label APL swapping and library disks.
11. Restore the distributed APL library.
12. Initiate APL.
13. Condense all libraries.

A complete example system generation is shown in Appendix B.

Step 1: Restore the distributed system tape to a system residence disk.

This procedure is described in C24-5033, DOS System Generation and Maintenance, (about page 20).

The distribution tape appears as follows:

```
IPL
Initialize disk program
Tape mark
IPL
Restore program
File identification record
File label information
APL and DOS System
Tape mark
Tape mark
```

This step consists of 2 parts: initializing the SYSRES disk or bypassing the initialization, and restoring the system to disk. BPS error messages are summarized in Appendix C.

BPS control card parameters.

```
yy   - year, 00 to 99 decimal.
ddd  - day of the year, 001 to 366 decimal.
cuu  - hexadecimal I/O channel and unit address.
Dd   - D1 for 2311 disks,
     - D3 for 2314 disks.
Tz   - T2 for 9 track tape,
     - T1,X'90' for 7 track tape.
```

Initialize the SYSRES volume.

A DOS SYSRES must have the VTOC on cylinder 199. Either the BPS Initialize Disk program supplied on the distributed system tape, or DASDI may be used. The following procedure is required when the BPS program is used:

a) Place the following cards in the card reader:

```
// JOB INTDSK
// DATE yyddd
// ASSGN SYSOPT,X'cuu',Dd  (Disk to be initialized)
// ASSGN SYSLOG,X'cuu',C1  (1052 printer keyboard)
// ASSGN SYSLST,X'cuu',L1  (Printer)
// EXEC
// UID IA
// VTOC STRTADR=(0199000),EXTENT=(10)
VOL1APLSYS
// END
```

b) Set the address of the tape drive containing the distributed system tape on the CPU load unit switches.

c) Press LOAD.

d) When the WAIT light comes on, press START and EOF on the card reader.

e) The message "EOJ" will be printed on the 1052 when the initialization is complete.

Alternative: Bypass SYSRES initialization.

If the disk has been previously initialized, the Initialize Disk function on the distributed system tape can be bypassed as follows:

a) Place the following cards in the card reader:

```
// JOB INTDSK
// DATE yyddd
// ASSGN SYSLOG,X'cuu',C1  (1052 printer keyboard)
// ASSGN SYSIPT,X'cuu',Tz (Distribution tape)
// FILES SYSIPT,1
```

b) Set the address of the system tape on the CPU load unit switches.

c) Press LOAD.

d) When the WAIT light comes on, press START and EOF on the card reader.

e) The tape will space, and the following message will appear on SYSLOG:

```
000C
4000A
```

Restore the distributed APL system to disk.

After initializing the disk, or bypassing the Initialize function:

a) Do not rewind the distribution tape.

b) Place the following cards in the card reader:

```
// JOB DISRST
// DATE yyddd
// ASSGN SYS000,X'cuu',Dd  (Initialized disk)
// ASSGN SYSLST,X'cuu',L1  (Printer)
// ASSGN SYSLOG,X'cuu',C1  (1052 printer keyboard)
// ASSGN SYS002,X'cuu',Tz   (2nd reel, 2314 sysgen only)
// EXEC
```

c) IPL the distributed system tape.

d) When the WAIT light comes on, press START and EOF on the card reader.

Step 2: IPL the restored system.

Since the distributed system contains no physical or logical unit blocks, these must be added during the IPL procedure with ADD and ASSGN statements. This procedure is described in both C24-5033, DOS System Generation and Maintenance, and in C24-5022, DOS Operating Guide.

Step 3: Initialize the SYSRES label cylinder.

The distributed system contains no standard labels, and these must be placed on SYSRES by the procedure described in C24-5033, DOS System Generation and Maintenance.

Step 4: Assemble and Catalog the DOS Supervisor

This step creates a DOS supervisor which will support APL on a particular system configuration. A program consisting of a sequence of macro calls is assembled to obtain a relocatable object deck. This deck is then link edited and catalogued in the Core Image Library.

DOS Supervisor Macros are described in detail in C24-5033, DOS System Generation and Maintenance. APL requires that the DOS supervisor have the specific parameter values shown in Table 2. Parameter values which are not shown may be specified as described in DOS System Generation and Maintenance.

| MACRO | REQUIRED PARAMETER VALUES |
|---|---|
| SUPVR | SYSTEM=DISK,<br>MPS=YES,<br>TP=APL,<br>MICR=NO |
| CONFG | SP=YES,<br>DEC=YES,<br>FP=YES,<br>TIMER=YES |
| STDJC | All parameters are optional |
| FOPT | IT=F1,<br>PC=YES,<br>CCHAIN=YES,<br>DASDFP=NO,<br>SKSEP=NO,<br>CE=NO |
| PIOCS | SELCH=YES,<br>BMPX=NO |
| ALLOC | See note a |
| IOTAB | All parameters are optional |
| DVCGEN | See note b |
| ASSGN | See note c |
| HOLD | See note d |
| SEND | X'2800' required for a 10k Supervisor. (A larger supervisor may be generated if desired.) |

Table 2: REQUIRED DOS SUPERVISOR PARAMETER VALUES

Notes to Table 2:

a. ALLOC F1=nk,F2=mK

Storage allocation is optional, and may be controlled in the usual way by the machine operator. (See C24-5022, DOS Operating Guide.) However, it will often ease APL initiation if the foreground 1 partition is pre-set to the size required. The calculations to determine the storage allocation required for APL are described in Step 6.

b. DVCGEN

DVCGEN calls for 2702 or 2703 lines are not required for APL, since these are included in the APL configuration.

c. ASSGN

The ASSGN macro has been modified so that logical units belonging to a foreground partition may be pre-assigned.

Example: ASSGN SYS004,X'170',F1

The partition (F1, F2 or BG) is included following the physical unit designation.

APL\360 library files must not be assigned to SYS004 or SYS005.

d. HOLD F1 (and/or HOLD F2)

This macro sets the "HOLD bit" in the appropriate PIB. This is particularly useful when logical units within foreground partitions are initially assigned, since the assignments will be held when the partition terminates or is cancelled. The "HOLD" can, of course, be overridden by the Machine Operator with a RELSE command. (See C24-5022, DOS Operating Guide.)

DOS Supervisor Assembly Diagnostics. These diagnostics following the Supervisor assembly may be ignored:

| | |
|---|---|
| IJQ046 | AT LEAST ONE RELOCATABLE Y-TYPE CONSTANT IN ASSEMBLY |
| IJQ037 | MNOTE STATEMENT |
| IJQ041 | UNDECLARED VARIABLE SYMBOL |

The DOS supervisor link edit. The relocatable object deck obtained from the assembly of the DOS Supervisor must be link edited as described in C24-5033, DOS System Generation and Maintenance.

Distributed DOS supervisor. The following deck was used to assemble the supervisor which is contained on the distributed system tape.

```
// EXEC ASSEMBLY
   SUPVR SYSTEM=DISK,MPS=YES,TP=APL
   CONFG MODEL=50,SP=YES,DEC=YES,FP=YES,TIMER=YES
   STDJC DUMP=NO
   FOPT  OC=YES,IT=F1,PC=YES,TEB=10,CCHAIN=YES,CE=NO
   PIOCS SELCH=YES,TAPE=7
   IOTAB BGPGR=25,F1PGR=10,JIB=20,CHANQ=25,IODEV=50
   SEND X'2800'
   END
/*
```

Step 5: The APL\360 Configuration

A program consisting of a sequence of configuration macros must be assembled to obtain a relocatable object deck, which must be catalogued in the Relocatable Library. This program tailors APL to a particular system by determining the following:

1. The device addresses of all 2702 or 2703 lines to be used by APL\360.

2. The amount of core storage required.

3. The size of the swap area on disk.

The APL configuration will normally assemble without error. If, however, an APLDEV statement for a device address greater than X'7F' has been included, the following ASSEMBLER diagnostic will appear spuriously:

DATA ITEM TOO LARGE

| OPERATION | OPERAND | COMMENT |
|---|---|---|
| APLSCONF | DIRS=d<br>(DIRS=11) | The number d should be prime. It specifies the number of user directories in the APL library. Each holds about 400 users and requires 21 tracks (2311) or 10 tracks (2314). |
| | INCORE=i<br>(INCORE=3) | Specifies the number of APL users in storage concurrently (minimum of 2). Each uses 36,864 bytes. |
| | COPIES=c<br>(COPIES=10) | Specifies the number of concurrent copy operations allowed; further requests are deferred. This value affects the amount of disk swapping area required. |
| APLDEV | X'uu' | APLDEV must be called once for each 2702 or 2703 address, in ascending order by device address. X'uu' is the device address of the 2702/3 line on the multiplexor channel. |
| | (TYPE=AMBIG)<br>TYPE=1050<br>TYPE=2741<br>TYPE=TS41 | Specifies the type of device on 2702/3 address X'uu'. TYPE=AMBIG is required for a dial-up device. For a fixed connection, the correct type must be specified as indicated (TS41 denotes a 2741 with PTTC/EBCD keyboard; 2741 denotes standard Selectric). |
| | SAD=SAD0<br>(SAD=SAD1)<br>SAD=SAD2<br>SAD=SAD3 | Specifies the terminal control address wired to the corresponding 2702 line. Field Engineers can supply this data. The SAD specified for a 2703 is irrelevant. |
| APLSEND | none | Must follow the last call of APLDEV and precede the END card. |

Note: A number must be substituted for each parameter shown as a lowercase letter. In other cases, all of the possible alternatives are shown. The default values appear in parentheses.

Table 3: APL CONFIGURATION MACROS

When a configuration deck is satisfactorily assembled, the relocatable object deck is catalogued in the Relocatable Library by the following job:

```
// JOB CATCONF
// EXEC MAINT
     relocatable object deck is placed here
/*
/&
```

Step 6: Estimating APL Swapping and Storage Requirements.

Swapping requirements. The disk area reserved for APL swapping must be large enough to contain the maximum number of active workspaces possible in a particular system configuration. This number is equal to:

The number of APLDEV calls, plus the value specified for COPIES, plus two, minus the value specified for INCORE. (See the parameters to the APLSCONF macro, Table 3.)

Each active workspace requires 10 2311 tracks or 5 2314 tracks. The total number of tracks required is then the number of workspace areas required, multiplied by the appropriate number of tracks per workspace.

Note: It is advisable to allow one or two extra workspaces, since a set of tracks for a workspace in the swapping area which contains a flagged bad track will be abandoned.

Core storage requirements. The APL supervisor and interpreter require 72000 bytes. In addition, 36864 bytes are required for each user in core (INCORE=c), and 600 bytes for each 2702 or 2703 line (i.e. APLDEV statement). The total must be rounded up to the next multiple of 2048 for storage protection block alignment. The value required in an alloc statement (see Step 4) is then the total storage required by APL divided by 1024.

The corresponding value specified in the PHASE APLSMPS statement (see Step 9) will be the machine core size in multiples of 1024 minus the storage required by APL.

## Step 7: Assemble and Catalog APL Disk Parameters

APL\360 determines the physical addresses, device types, and extents of its swapping and library files during initialization. Two items of information are required for each file: the logical unit number and the text of the Format 1 label. This information is provided by a relocatable object deck obtained by assembling a sequence of disk parameter macros. The object deck must be catalogued in the Relocatable Library.

The sequence of APL disk parameter macros has the following form:

```
SWAP    APLDS   n,'swap file label'       (note a)
LIB     APLDS   o,'first library label'   (note b)
        APLDS   p,'second library label'  (note c)
        .
        .
        APLDISK                           (note d)
        END
```

Notes:

In the above example, n, o and p denote the logical unit numbers to which the corresponding devices will be assigned prior to APL\360 initiation. The quoted field corresponds to the text of the Format 1 label which will be written in the VTOC of the corresponding disk pack. The text of this information must contain no more than 44 characters, and must be identical to the text of the Format 1 labels written on the disk pack which will be assigned to the specified logical unit.

a. The first call of APLDS must have the identifier SWAP, and the text must correspond to the label written on the swapping pack. Only one swap file is allowed.

b. The second call of APLDS must have the identifier LIB. The APL\360 user directories are written at the beginning of this file. It is advisable to place this file on a different physical device from the swap file.

Step 7 Notes, cont.

c. When the APL library becomes too large for one pack, additional calls of APLDS are included following the LIB APLDS statement to define additional files. These additional calls must not have identifiers. The number of files containing the APL\360 library is limited to 20.

d. The APLDISK macro must follow the last call of APLDS.

When the disk parameters have been successfully assembled, the relocatable object deck is catalogued in the Relocatable Library by the following job:

```
// JOB DPARS
// EXEC MAINT
relocatable object deck goes here
/*
/&
```

## Step 8: Linkage Edit the APL Utility Program

The APL\360 Utility program can now be link edited as follows:

```
// JOB UTILLINK  LINK EDIT APL UTILITY
// OPTION LINK,CATAL
 INCLUDE APLSUTIL
// EXEC LNKEDT
/&
```

## Step 9: Linkage Edit APL

The following job will linkage edit APL and catalogue it in the Core Image Library:

```
// JOB APLLINK    LINK EDIT APL
// EXEC MAINT
 DELETC APLS.ALL
 CONDS CL
/*
// OPTION LINK,CATAL
 INCLUDE
 PHASE APLSMPS,F+xxK        see note
 INCLUDE APLSLINK
/*
// EXEC LNKEDT
/&
```

Note: The statement PHASE APLSMPS,F+xxK names the core image phase of APL and specifies the location in storage where it will be loaded. The value to be substituted for xx should have been determined in Step 6 (Estimating APL Swapping and Storage Requirements).

APL Link Edit Messages. These diagnostics following the APL storage map may be ignored:

```
* UNREFERENCED SYMBOLS
      POSSIBLE INVALID ENTRY POINT DUPLICATION IN INPUT
      CONTROL SECTIONS OF ZERO LENGTH IN INPUT
      011 UNRESOLVED ADDRESS CONSTANTS  (The number may vary)
```

## Step 10: Label APL Swapping and Library Disks

During APL Initialization, the foreground 1 logical units specified in the disk parameter assembly are searched for the associated Format 1 labels. These labels are written on the proper packs by a program in the Relocatable Library, and which is executed as follows:

a) 2311 Systems

```
// JOB DASDLBL
// OPTION LINK,CATAL
 INCLUDE APL2311
// EXEC LNKEDT
/&
```

b) 2314 Systems

```
// JOB DASDLBL
// OPTION LINK,CATAL
 INCLUDE APL2314
// EXEC LNKEDT
/&
```

A set of control cards of the following form is required for each label to be written:

```
// ASSGN SYSnnn,X'cuu'
// DLBL LBLDTF,'file identification',99/365
// EXTENT SYSnnn,serial,1,1,first-track,number-of-tracks
// EXEC DASDILBL
```

The DLBL and EXTENT statements are described in detail in C24-5022, DOS Operating Guide.

Notes:

1. SYSnnn on an EXTENT statement must be the same as SYSnnn on the preceding ASSGN statement.

Step 10 Notes, cont.

2. The first-track parameter of the EXTENT statement must correspond to track zero of some cylinder. (Tracks are counted from cylinder zero, track zero. Cylinder 150, track 0 on a 2311 is track 1500, on a 2314 it is track 3000. In general, the track number is computed from the track address by the formula $TN \leftarrow H + C \times N$, where $TN$ is the track number, $H$ is the head part of the track address, $C$ is the cylinder, and $N=10$ for 2311's and $N=20$ for 2314's.)

3. The number-of-tracks parameter of the EXTENT statement must represent an integral number of cylinders.

Step 11: Restore the Distributed APL Library

The APL utility program is used to create the initial set of library directories and to restore the distributed library to the library disks as follows:

Mount the distributed library tape and then run the following job in the background partition:

```
// JOB CREATE APL LIBRARY
ALLOC F1=0K,F2=0K
// ASSGN SYS004,X'cuu'     Distributed library tape.
// ASSGN SYSnnn,X'cuu'     Library disks.
.
.
//EXEC APLUTIL
CREATE
ACCTG 1
/*
ALLOC F1=nK,F2=mK     (Standard settings from step 4)
/&
```

Note: The long accounting operation is not required (see APL\360: Library Maintenance) but will list all workspaces contained on the library disks, verifying that the directories have been properly created and the distributed library restored.

When this job completes successfully, the distributed library has been restored to disk. For details of the contents of this library, see APL\360: User's Manual, Part 4.

Step 12: Initate APL.

When this point is reached, a complete APL system generation has been performed. The newly generated system should now be tested by initiating APL\360 with a sequence of SYSLOG operations of the following form:

```
BG alloc f1=214K  (amount determined in Step 6)
BG stop
(press 1052 request)
AR 1160A READY FOR COMMUNICATIONS
AR start f1
F1 assgn sysnnn,x'cuu'
F1 exec aplsmps
```

Note: One assgn statement is required for each logical unit specified in the disk parameters assembly. Both the logical unit assignments and the storage allocation can be preset when the DOS supervisor is assembled. If this has been correctly done and the DOS operator has not changed any of the specifications, then the alloc and assgn commands are not required.

nnn is the logical unit number specified in the APLDS call, and

cuu is the channel and unit address of the disk on which the pack bearing the associated Format 1 label is mounted.

If the storage allocation and load address are correct, APL will load into the foreground 1 partition, initialize the swap area, and wait for the APL operator to sign on. The significance of messages printed on SYSLOG during APL initialization is described in APL\360: Operator's Manual, Appendix A. Several seconds will elapse after the statement exec aplsmps until the message APL IS RUNNING is printed. During this time, the swapping disk is being initialized and checked for bad tracks.

At this point, the System/360 should appear idle except for the interval timer (seen on the console of some models) which should be positive and cycling. The APL Operator should attempt to sign on (after establishing a connection if his terminal is dial-up) by entering:

```
)SIN159
```

If the newly generated system is operating properly, APL\360 will acknowledge the sign-on of the operator as follows:

```
)ERR)  time  date      RAL R
```

The keyboard of the APL 360 operator's terminal remains locked unless attention is signalled.

The APL Operator may now begin adding users to the system. If the APL Operator can successfully sign on the system has probably been correctly generated.

A small test function contained in the workspace OPENC can be executed from the operator's terminal as follows:

```
      ) [illegible] OPENC
      [illegible]  time  date
      [illegible]
```

This defined function will give the operator a series of instructions, allowing him to initialize the workspace [illegible] which provides several utility functions.

The APL Operator should examine the contents of public library 1, to which the distribution library should have been restored. He can do this with the command:

```
      )LIB 1
```

which will cause a list of the workspaces in public library 1 to be printed.

Some common problems:

The system enters a tight loop immediately after the message APL IS RUNNING.

- Check Step 11.
- Have a Customer Engineer verify that the proper condition code is being set after a SIO instruction to the Transmission Control Unit.
- If the Transmission Control is control unit X'80' or above, verify that the System/360 has extended bump storage.

The telephones attached to the Transmission Control Unit will not answer.

- Ensure that the data-sets have power and are in the auto-answer mode.
- Verify that the proper SAD commands were specified in Step 5.

A 2741 keyboard repeatedly unlocks immediately after the APL operator enters [illegible].

- Ensure that the first character entered is on the APL keyboard.
- Verify that the Transmission Control is at a recent Engineering Change level.
- Try another 2741.

Step 13: Condense All Libraries

At this point, a backup copy of the SYSRES disk should be created.

If separate maintenance and operational SYSRES packs are to be maintained, the modules listed in Appendix A may be deleted from the operational pack. The following job will delete the label writing program from the Core Image Library and condense all libraries:

```
// JOB CONDENSE ALL LIBRARIES
// EXEC MAINT
     DELETC DASDILBL
     CONDS CL,RL,SL
/*
/&
```

## SOME SYSTEM MAINTENANCE NOTES

DOS Maintenance. DOS may be maintained in the usual fashion except that:

DOS SYSTEM GENERATION MACROS MAY NOT BE REPLACED WITH STANDARD DOS UPDATES. THEY MUST BE REPLACED WITH DOS/APL UPDATES.

APL Library Disks. The number and organization of APL library disks may be changed at any time by the following procedure:

a) Use the APL Utility Program to DUMP all libraries to tape.

b) Use DASDILBL to relabel the library and/or swap packs (This must be done to change extents).
- See system generation Step 10.

c) If the number of packs or data-sets is to be changed, perform system generation Steps 7, 8, 9, and 10.

d) DISKFMT all library packs, and RESTORE the tape dumped in step a) above.

The number of directories may be changed as follows:

a) DUMP all libraries to tape.

b) Perform system generation Steps 5, 8, and 9. Or,

If the library disks are to be reorganized as well, perform system generation Steps 5, 7, 8, 9, and 10.

c) Restore the tapes dumped in Step a) above by using a CREATE Utility program operation.

## APL\360 LIBRARY MAINTENANCE

Off-line maintenance of the APL library is performed by the APL Utility Program, which provides the following functions:

Library disk formatting.

Copying workspaces from disk to tape.

Copying workspaces from tape to disk.

Printing accounting information.

Creating APL library disks.

The major APL Utility error messages are given in Appendix D.

Note: The APL Utility Program must not be used while APL is running.

APL Utility I/O assignments. SYS004 and SYS005 must be assigned to the unit(s) on which dump or restore tapes will be mounted. The Utility opens SYS004 at the beginning of each function requiring magnetic tape, and alternates between SYS004 and SYS005 at end-of-volume.

All APL library disks must be assigned to the logical unit numbers specified in the APL Disk Parameters assembly (see system generation Steps 7 through 11.)

// ASSGN cards required by the Utility on a specific configuration may be found in Appendix B, Step 11.

APL Utility Program control cards. Utility functions are specified by control cards (shown in Table 4) which contain a function name and perhaps a parameter. Parameters may not be omitted. The function name and parameter, if any, must be separated by at least one blank, but may appear anywhere on the card.

APL UTILITY FUNCTIONS

DISKFMT - Format a library disk. A library disk must be formatted (after it has been labelled) before APL libraries can be written on it. The parameter, a number, specifies which disk is to be formatted. The library APLDS statements in the APL Disk Parameters assembly (see system generation Step 7), are numbered zero through k-1, beginning with the call labelled LIB.

Note: DISKFMT will destroy all information in the extent to be formatted.

DUMP - Dump all libraries to tape. All workspaces are written on tape, preceded by the library directories. All accounting information and information pertaining to workspaces is retained. The tape to be written on is first checked for unexpired labels, and if the file is accepted, header labels are written. At end-of-volume and end-of-file, the file name (APL LIBRARY DUMP), reel sequence number, and creation date are printed on SYSLOG.

The parameter required on the DUMP control card specifies the maximum tape record length. The recommended value is 10000. The minimum value is 500.

SELDUMP - Dump selected workspaces to tape. The selective dump operation writes on tape only those workspaces which are selected by cards which follow the SELDUMP statement and which precede an END statement. These cards have the following form:

account wsname

account - library number, right justified in card columns 1 - 10.

wsname - name of the workspace to be selected, beginning in card column 16. If the workspace name is omitted, all workspaces from the selected library are dumped. The last workspace selection card must be followed by an END card.

The parameter which must be specified on the SELDUMP control card specifies the maximum tape record length. The recommended maximum length is 10000.

RESTORE - Restore libraries to disk. Libraries written on tape by a DUMP operation are restored to the library disks, completely replacing their contents. If the tapes were dumped by a system with a different library configuration that that which is to restore them, a CREATE operation may be required.

Since the available storage on the library disks becomes fragmented, it is desirable to perform a DUMP followed by a RESTORE ocassionally to condense the active storage and consolidate the available space.

SELREST - Store workspaces from tape. Unlike a RESTORE function, SELREST does not completely replace the library disks. Rather, the workspaces read from tape are stored in libraries which already exist on the disks, in a manner analogous to the *)SAVE* command, to the library numbers from which they were dumped. A workspace restored to a library which already contains a workspace with the same name will replace it. The tapes restored by SELREST operations are usually obtained by a selective dump. This pair of functions is useful for exchanging workspaces between APL systems and for recovering lost workspaces from backup disks.

ACCTG - Print accounting. The parameter must be 0 (short accounting) or 1 (long accounting).

A short accounting prints a summary of cumulative connect time, cumulative CPU time, total workspaces saved, tracks occupied by workspaces, and the quota for each user. In addition to this information, the long accounting prints the name, account number of the saver, and disk location of all workspaces.

CREATE - Create an APL library. The CREATE function is required when the number of directories contained on a library dump tape does not agree with the number expected by the system, and is normally used to restore the APL libraries after the number of directories has been changed.

The CREATE function formats all library packs, converts the directories on tape to a proper set on disk, and then restores the workspaces.

VERIFY - Verify library disk condition. The parameter specifies a library file in the same manner as the parameter to DISKFMT. The specified library file is checked, and messages are logged on SYSLOG when tracks in error are encountered.

| FUNCTION | PARAMETER | COMMENT |
|---|---|---|
| DISKFMT | lib | Format the specified library. |
| DUMP | block | Dump all workspaces to tape. |
| SELDUMP | block | Dump specified workspaces to tape. |
| RESTORE | | Restore all libraries from tape. |
| SELREST | | Restore a SELDUMP tape to disk. |
| END | | Terminate SELDUMP. |
| ACCTG | 0 or 1 | Print user accounting information. |
| CREATE | | Create APL library disks. |
| VERIFY | lib | Verify library disk condition. |

Table 4: APL UTILITY CONTROL CARDS

APPENDIX A

APL SYSTEM GENERATION MODULES

The following job will delete all APL system generation modules from the Relocatable and Source Statement Libraries.

```
// JOB DELETE APL SYSGEN MODULES
// EXEC MAINT
     DELETR APL.ALL
     DELETS A.CDCOMP
     DELETS A.APLDEV
     DELETS A.ZSYMBOLS
     DELETS A.TRCOMP
     DELETS A.APLDEFN
     DELETS A.PERTERM
     DELETS A.DIRSECT
     DELETS A.APLSCONF
     DELETS A.MDVS
     DELETS A.APLSYS
     DELETS A.DEL
     DELETS A.MAKFR
     DELETS A.DIVTAB
     DELETS A.MPUB
     DELETS A.PCGEN
     DELETS A.PDGEN
     DELETS A.APLSEND
     DELETS A.APLDS
     CONDS RL,SL
     /*
/&
```

## APPENDIX B

## EXAMPLE SYSTEM GENERATION

Configuration: System/360 Model 50 H

| Device | Address |
|---|---|
| 1052 Console | 009 |
| 2540 Reader | 00C |
| 2540 Punch | 00D |
| 1403 Printer | 00E |
| 2702 Transmission Control: | |
| IBM 4 wire modems, SAD 0 | 020 to 02F |
| 103A2 Dial up, SAD 1 | 030 to 03E |
| 2314 Direct Access Storage | 170 to 177 |
| 2400 9 Track Tape | 282 and 283 |

Step 1: Restore Distributed System Tape.

Tapes mounted on 282 and 283, disk for SYSRES on 170.

Cards in reader:

```
// JOB INTDSK     INITIALIZE DISK FUNCTION
// DATE 68092
// ASSGN SYSLOG,X'009',C1
// ASSGN SYSLST,X'00E',L1
// ASSGN SYSOPT,X'170',D3
// EXEC
// UID IA
// VTOC STRTADR=(0199000),EXTENT=(10)
VOL1APLSYS
// END
```

Interval Timer disabled, IPL 282, ready reader (push START and EOF).

Restore Function, cards in reader:

```
// JOB DISRST
// DATE 68092
// ASSGN SYSLOG,X'009',C1
// ASSGN SYSLST,X'00E',L1
// ASSGN SYS000,X'170',D3
// ASSGN SYS002,X'283',T2
// EXEC
```

Interval Timer disabled, IPL 282, ready reader ( push START and EOF).

Step 2: IPL the Restored System

Cards in reader:

```
ADD X'009',1050A
ADD X'00C',2540R
ADD X'00D',2540P
ADD X'00E',1403
ADD X'170',2314
ADD X'171',2314
SET DATE=04/01/68,CLOCK=00/00/00
ASSGN SYSLOG,X'009'
ASSGN SYSLST,X'00E'
ASSGN SYSIN,X'00C'
ASSGN SYSPCH,X'00D'
ASSGN SYSLNK,X'171'
ASSGN SYS001,X'171'
ASSGN SYS002,X'171'
ASSGN SYS003,x'171'
```

IPL 170, ready reader (press START and EOF).

Step 3: Initialize the SYSRES Label Cylinder

Standard labels for work files:

```
// JOB STDLABEL
// OPTION STDLABEL
// DLBL IJSYSLN,'APLDOS LINK FILE',99/365
// EXTENT SYSLNK,333333,1,0,20,240
// DLBL IJSYS01,'APLDOS UTILITY FILE 1',99/365
// EXTENT SYS001,333333,8,0,260,500,9
// DLBL IJSYS02,'APLDOS UTILITY FILE 2',99/365
// EXTENT SYS002,333333,8,0,270,500,19
// DLBL IJSYS03,'APLDOS UTILITY FILE 3',99/365
// EXTENT SYS003,333333,8,0,1260,500,9
/&
```

Step 4: Assemble and Catalogue a new DOS Supervisor

```
// JOB DOSSUP   ASSEMBLE DOS SUPERVISOR
// OPTION LIST,DECK,XREF
// EXEC ASSEMBLY
     SUPVR SYSTEM=DISK,MPS=YES,TP=APL
     CONFG MODEL=50,SP=YES,DEC=YES,FP=YES,TIMER=YES
     STDJC DUMP=NO
     FOPT  OC=YES,IT=F1,PC=YES,TEB=10,CCHAIN=YES,CF=NO
     PIOCS SELCH=YES,TAPE=7
     ALLOC F1=210K,F2=0K
     IOTAB BGPGR=25,F1PGR=10,F2PGR=5,JIB=20  HANO=15,HODEV=2
     PRINT NOGEN
     DVCGEN CHUN=X'009',DVCTYP=1050A
     DVCGEN CHUN=X'00C',DVCTYP=2540R
     DVCGEN CHUN=X'00D',DVCTYP=2540P
     DVCGEN CHUN=X'00E',DVCTYP=1403
     DVCGEN CHUN=X'170',DVCTYP=2314
     DVCGEN CHUN=X'171',DVCTYP=2314
     DVCGEN CHUN=X'172',DVCTYP=2314
     DVCGEN CHUN=X'173',DVCTYP=2314
     DVCGEN CHUN=X'174',DVCTYP=2314
     DVCGEN CHUN=X'175',DVCTYP=2314
     DVCGEN CHUN=X'176',DVCTYP=2314
     DVCGEN CHUN=X'177',DVCTYP=2314
     DVCGEN CHUN=X'282',DVCTYP=2400T9
     DVCGEN CHUN=X'283',DVCTYP=2400T9
     ASSGN SYSLOG,X'009'
     ASSGN SYSRDR,X'00C'
     ASSGN SYSIPT,X'00C'
     ASSGN SYSPCH,X'00D'
     ASSGN SYSLST,X'00E'
     ASSGN SYSLNK,X'171'
     ASSGN SYS001,X'171'
     ASSGN SYS002,X'171'
     ASSGN SYS003,X'171'
     ASSGN SYS006,X'172',F1   APL SWAP FILE
     ASSGN SYS007,X'173',F1   APL LIB 0
     ASSGN SYS008,X'172',F1   APL LIB 1
     HOLD  F1
     PRINT GEN
     SEND  X'2800'
     END
/*
/&
// JOB DOS/APL    CATALOGUE NEW SUPERVISOR
// OPTION LINK,CATAL
     INCLUDE
Relocatable object from preceding assembly placed here.
/*
// EXEC LNKEDT
/&
```



Step 5: Assemble and Catalogue the APL Configuration.

```
// JOB CONFIG   ASSEMBLE APL CONFIGURATION
// EXEC ASSEMBLY
CONF      TITLE 'APL CONFIGURATION'
          PRINT  NOGEN
          APLSCONF    ,  DEFAULT OPTIONS
          APLDEV X'21',TYPE=2741,SAD=SAD0
          APLDEV X'22',TYPE=2741,SAD=SAD0
          APLDEV X'23',TYPE=2741,SAD=SAD0
          APLDEV X'24',TYPE=2741,SAD=SAD0
          APLDEV X'25',TYPE=2741,SAD=SAD0
          APLDEV X'26',TYPE=2741,SAD=SAD0
          APLDEV X'27',TYPE=2741,SAD=SAD0
          APLDEV X'28',TYPE=2741,SAD=SAD0
          APLDEV X'29',TYPE=2741,SAD=SAD0
          APLDEV X'2A',TYPE=TS41,SAD=SAD0
          APLDEV X'2B',TYPE=TS41,SAD=SAD0
          APLDEV X'2C',TYPE=TS41,SAD=SAD0
          APLDEV X'2D',TYPE=TS41,SAD=SAD0
          APLDEV X'2E',TYPE=TS41,SAD=SAD0
          APLDEV X'2F',TYPE=TS41,SAD=SAD0
          APLDEV X'30'       DEFAULT OPTIONS
          APLDEV X'31'
          APLDEV X'32'
          APLDEV X'33'
          APLDEV X'34'
          APLDEV X'35'
          APLDEV X'36'
          APLDEV X'37'
          APLDEV X'38'
          APLDEV X'39'
          APLDEV X'3A'
          APLDEV X'3B'
          APLDEV X'3C'
          APLDEV X'3D'
          APLDEV X'3E'
          APLSEND
          END
/*
/&

// JOB CATCONF   CATALOGUE APL CONFIGURATION
// EXEC MAINT
Relocatable object from preceding assembly placed here.
/*
/&
```

Step 6: Estimate APL swapping and storage requirements

```
      INCORE←3
      COPIES←10
      LINES←30
      CORE←2048×⌈(72000+(INCORE×36864)+LINES×600)÷2048
      CORE
100704
      CORE:1024
196
      LOADADDRESS←256-196
      LOADADDRESS
60
      WORKSPACES←LINES+COPIES+2-INCORE
      WORKSPACES
30
      TRACKS←5×WORKSPACES
      TRACKS
195
```

Required allocation: alloc f1=196k
Phase address: PHASE APLSMPS,F+60K
Reserve 11 2314 cylinders for swapping.

Step 7: Assemble and Catalogue APL Disk Parameters.

```
// JOB DISKPARS   ASSEMBLE APL DISK PARAMETERS
// EXEC ASSEMBLY
PARS      TITLE 'APL DISK PARAMETERS'
SWAP      APLDS 6,'APL SWAP FILE'
LIB       APLDS 7,'APL LIB ZERO'
          APLDS 8,'APL LIB ONE'
          APLDISK
          END
/*
/&
// JOB CATPARS   CATALOGUE DISK PARAMETERS
// EXEC MAINT
Relocatable object deck from preceding assembly placed here.
/*
/&
```

Step 8: Link Edit the APL Utility Program

```
// JOB UTILLINK   LINK EDIT APL UTILITY
// OPTION LINK,CATAL
     INCLUDE APLSUTIL
// EXEC LNKEDT
/&
```

Step 9: Link Edit APL.

```
// JOB APLLINK     LINK EDIT APL
// EXEC MAINT
     DELETC APLS.ALL
     CONDS CL
/*
// OPTION LINK,CATAL
     INCLUDE
     PHASE APLSMPS,F+60K
     INCLUDE APLSLINK
/*
// EXEC LNKEDT
/&
```

Step 10: Label APL Swap and Library Disks

Initialized disks with serials APL001 and APL002 mounted on 172 and 173

```
// JOB DASDILBL   LINK EDIT LABEL WRITING PROGRAM
// OPTION LINK,CATAL
     INCLUDE APLL2314
// EXEC LNKEDT
/&

// JOB LABELS   LABEL DISK PACKS
// ASSGN SYS006,x'172'
// DLBL LBLDTF,'APL SWAP FILE',99/365
// EXTENT SYS006,APL001,1,1,20,220
// EXEC DASDILBL
// ASSGN SYS007,X'173'
// DLBL LBLDTF,'APL LIB ZERO',99/365
// EXTENT SYS007,APL002,1,1,20,3960
// EXEC DASDILBL
// ASSGN SYS008,X'172'
// DLBL LBLDTF,'APL LIB ONE',99/365
// EXTENT SYS008,APL001,1,1,240,3740
// EXEC DASDILBL
/&
```

Step 11: Restore the Distributed APL Library

Distributed Library tape mounted on 283.

```
// JOB RESTORE DISTRIBUTION LIBRARY
// ASSGN SYS004,X'283'
// ASSGN SYS007,X'173'
// ASSGN SYS008,X'172'
ALLOC F1=0K
// EXEC APLUTIL
CREATE
/*
ALLOC F1=196K
/&
```

Step 13: Condense All Libraries.

An operational SYSRES volume is to be created. A separate maintenance pack will not be maintained. Instead, the APL System Generation modules will be saved on tape, to be restored when APL maintenance is required.

```
// JOB PUNCH APL SYSGEN MODULES
// ASSGN SYSPCH,X'282'  TAPE FOR SYSGEN MODULES
// EXEC SSERV
     PUNCH A.CDCOMP,A.APLDEV,A.ZSYMBOLS,A.TRCOMP
     PUNCH A.APLDEFN,A.PERTERM,A.DIRSECT,A.APLSCONF
     PUNCH A.MDVS,A.APLSYS,A.DEL
     PUNCH A.MAKFR,A.DIVTAB,A.MPUB,A.PCGEN,A.PDGEN
     PUNCH A.APLSEND,A.APLDS
/*
// EXEC RSERV
     PUNCH APL.ALL
/*
// CLOSE SYSPCH,X'00D'
/&
```

```
// JOB DELETE APL SYSGEN MODULES
// EXEC MAINT
     DELETS A.CDCOMP,A.APLDEV,A.ZSYMBOLS,A.TRCOMP
     DELETS A.APLDEFN,A.PERTERM,A.DIRSECT,A.APLSCONF
     DELETS A.MDVS,A.APLSYS,A.DEL
     DELETS A.MAKFR,A.DIVTAB,A.MPUB,A.PCGEN,A.PDGEN
     DELETS A.APLSEND,A.APLDS
     DELETR APL.ALL
     DELETC DASDILBL
     CONDS CL,RL,SL
/*
/&
```

At this point, the operational SYSRES pack is dumped to tape in order to provide a backup copy.



The APL System Generation modules can be restored as follows:

Tape punched above is mounted on X'282'

```
// JOB RESTORE APL SYSGEN MODULES.
// ASSGN SYSIPT,X'282'
// EXEC MAINT
// ASSGN SYSIPT,X'00C'
// ASSGN SYS000,X'282'
// MTC RUN,SYS000
/*
/&
```

## APPENDIX C

## BPS UTILITY MESSAGES

Where no reply is given, either none is required or none is allowed. Replies are entered by typing the reply character twice and then pressing EXTERNAL INTERRUPT.

| | |
|---|---|
| 0901 | Program Check |
| 0701 | Program Check |
| 0702A | 1052 request, Reply 4 to continue |
| 0cuu | Device I/O error, Followed by a line of sense information.<br>Reply 5 to retry.<br>Reply 4 to bypass and continue.<br>Reply 0 or 1 to terminate. |
| 1050A | Missing DATE card. |
| 1040A | Missing JOB card. |
| 1110A | Duplicate JOB card. |
| 1200A | Unrecognizable card. |
| 1220A | Invalid. or missing control card. |
| 13xxA | Invalid field (xx - field number) |
| 1703A | PAUSE card, Reply character - space. |
| 3001 | All disk files transferred to tape. |
| 3002 | The job name on the JOB card is not DISCPY |
| 3003 | Unrecoverable tape error. |
| 3012 | SYS001 and SYS002 must be assigned to tape. |
| 3015 | Tape convert check. |
| 3016 | SYS000 must be assigned to a disk. |
| 3022A | Error in // U card, Correct card and reply 2 to continue. |
| 3LCA1A | // U card missing., Correct and reply 2 to continue |
| 3741A | Invalid tape label, Reply 2 to delete label and continue. |
| 3742A | See 3741A |

BPS Utility messages, cont.

| | |
|---|---|
| 3751A | Active label found., Reply 2 to delete label and continue. |
| 3752A | See 3751A |
| 3688 | End of volume, switching to alternate. |
| 3888A | End of volume, Mount another reel and reply 2 to continue. |
| 3999 | SYSLST or SYSLOG not assigned, or assigned to the same device. |
| 4304 | Disk pack has no VOL1 label. |
| 4310A | Disk error., Reply 2 to retry, other to terminate. |
| 4341 | More than 1 format 1 label found for file. |
| 4343 | Record size greater than 4000 bytes. |
| 4401A | No format 1 label, Correct // U card and reply 2 to continue. |
| 3LC2A | Invalid VTOC start address or EXTENT |
| 3LC3A | Assigned VTOC overflows cylinder. |
| 3LC4A | VOL1 missing, or VOL1 through VOLn out of sequence. |
| 3LC5A | VOL1 card has blanks in volume serial field. |
| 3LC6A | Wrong VTOC or END card, or missing END card. |
| 3LC7A | VTOC card sets and number of assigned packs are not equal. |
| 3LC8A | Comma or blank must follow parameter. |
| 4404A | No Format 4 label, Reply 4 to ignore, 0 or 1 to terminate. |
| 4444A | Overlap on unexpired file - See 4404A. |
| 4306A | No VOL1 label, Replace pack and reply 2 to continue. |
| 4311A | Disk error, See 4306A. |
| 4400A | No space in VTOC |
| 4409A | No format 1 label |
| 4410A | Data check in output count field. |
| 4414A | Error while reading format 4 label. |
| 4419A | Read error while searching VTOC |
| 4428 | VTOC not on cylinder 199 |

## APPENDIX D

## APL UTILITY ERROR MESSAGES

UNIT NOT ASSIGNED, SYSnnn

Reason: A library disk is not assigned.
Action: The Utility program terminates.
Correction: Correct // ASSGN cards and resubmit.

WRONG DEVICE TYPE, SYSnnn

Reason: SYSnnn, which should be assigned to an APL library disk, is assigned to some other type of device.
Action: The Utility terminates.
Correction: Correct // ASSGN cards and resubmit.

ERROR IN DS LABEL, SYSnnn

Reason: The disk to which SYSnnn is assigned does not contain the label defined by the APLDS nnn,'label' statement.
Action: The Utility terminates.
Correction: Check // ASSGN cards, the disks mounted, and system generation Steps 7 through 10.

INCORRECT CONTROL CARD

Reason: The card which was printed on SYSLOG preceding this message was unrecognizable by the Utility.
Action: The Utility terminates.
Correction: Correct the card or card sequence and resubmit.

LOST saver lib name disk-location

Reason: The workspace named cannot be read from the library disks.
Action: The Utility continues.
Correction: None is possible.



INCORRECT NUMBER OF DIRECTORIES ON TAPE

Reason: A tape is mounted for a RESTORE function which was not dumped by the Utility currently operating.
Action: The RESTORE function terminates.
Correction: Check system generation Steps 5 through 8. If the system is correct, use the CREATE function.

FIRST WS IS NOT A DIRECTORY

Reason: The tape mounted for a RESTORE function is not the first reel of a tape written by a DUMP function.
Action: The RESTORE function terminates.
Correction: Mount the correct tape and resubmit.

NUMBER NOT FOUND wsid
WSNAME NOT FOUND wsid

Reason: A selection card during a SELDUMP function did not specify an existing workspace.
Action: The selection card is bypassed.
Correction: Correct card and resubmit.

LIBRARY NOT FOUND number

Reason: There is no library with the number of a workspace read from tape during a restore function.
Action: The workspace is bypassed.
Correction: The APL Operator must add the library number.

NOT AN APL DUMP TAPE

Reason: A tape mounted for a restore function does not contain an APL dump header label.
Action: The function terminates.
Correction: Mount the correct tape and resubmit.

UNEXPIRED FILE

Reason: An output tape mounted for a dump function contains an unexpired label (which is printed above this message).
Action: The Utility waits for an operator response from SYSLOG.
Correction: Reply ignore to write on the tape, or
reply cancel to terminate, or
mount a new reel and reply newtape.