PL/I BULLETIN No. 1

January 1966

CONTENTS

The PL/I Bulletin is sponsored by Working Group 4 (WG4) of the Special Interest Group on Programming Languages (SIGPLAN) of the Los Angeles Chapter of the Association for Computing Machinery.

Reproduction of the PL/I Bulletin is being provided by Logicon, Inc., Redondo Beach, California.

Distribution of the PL/I Bulletin is being provided by the Business Equipment Manufacturers Association (BEMA).

All inquiries and contributions should be addressed to:

R. N. Southworth (Editor, PL/I Bulletin)
c/o Logicon, Inc., 205 Avenue I
Redondo Beach, California 90277

PB1.0 EDITOR'S NOTES

PB1.0.1 About the PL/I Bulletin

The PL/I Bulletin is intended to be an informal publication for the interchange of information and opinions relating to PL/I. It is expected that the contents will be primarily of interest to PL/I specialists or language specialists. The contents will be divided into several sections, as indicated by the table of contents on page zero. The section headings for some sections are self-explanatory: "EDITOR'S NOTES", "REFERENCES", "CIRCULATION LIST", "CORRESPONDENCE", and "NEWS ITEMS".

Three sections, however, deserve some special description of the intended content: "WORKING PAPERS", "PROGRAMS" and "PITFALLS & PRAGMATISMS". The WORKING PAPERS section is to be a place for informal publication of technical papers. Presumably, publication in the PL/I Bulletin would not preclude later publication in a journal or magazine. Hopefully, the waiting time for publication in the PL/I Bulletin will not be as long as it is for most journals, so that quick reactions can be got from new concepts. Very little editing will be performed in working papers and, as long as space permits, judgment as to content will not be supercritical. Papers which are obviously intended for sales promotion will be rejected.

The PROGRAMS section is intended to be a place to publish programs and parts of programs written in PL/I language. Contributions may be a group, a block, several blocks or a complete program with input/output statements. It would be appreciated if contributions for the PROGRAMS section can be prepared in accordance with the Publication Language Guidelines published in Section PB1.3.1 of this Bulletin (or later revisions thereof). Editing of programs to be published will be minimal. Selection of programs will be performed by a review board of WG4 members.

The intended contents of the PITFALLS & PRAGMATISMS section is indicated by the title. Discussions of pitfalls will be brief and to the point, and contributions may be heavily edited to maintain continuity of style and format. Pragmatisms will be demonstrations of ways of using PL/I; ways which are novel and unusual as well as practical and useful.

PB1.0.2 About the Sponsor

The purpose of Working Group 4 (WG4) of the Special Interest Group on Programming Languages (SIGPLAN) of the Los Angeles Chapter of the Association for Computing Machinery is twofold:

1) to study, to evaluate and, in other ways, to acquire knowledge relating to the programming language known as PL/I (previously known as NPL),

2) to communicate knowledge relating to PL/I, within SIGPLAN and to other interested persons, by means of meetings, seminars and publications.

This statement of purpose is consistent with the statement of purpose of the L.A. SIGPLAN given in their bylaws: "Article I, Section B, Purpose. The purpose shall be to promote the acquisition and exchange of information on the theory, design, implementation, and application of programming languages in order to educate its members and advance the state of the programming art."

WG4 was conceived in late 1964 and first announced in the SIGPLAN "Notices" in December 1964. The decision to sponsor the PL/I Bulletin was made in October, 1965. The group felt that this was one of the most fruitful ways in which they could fulfill their purpose.

PB1.0.3 Solicited Correspondence

Your editor wrote to a number of computer specialists requesting them to submit their views regarding PL/I and its future for publication. The majority of the responders (for various reasons) preferred not to make any statement. Responses from those who did state their views are presented in the CORRESPONDENCE section with the heading "PL/I and Its Future".

PB1.1 NEWS ITEMS

PB1.1.1 The European Computer Manufacturers Association (ECMA) has formed a PL/I committee, TC10. The scope and program of work of TC10 are given below.

SCOPE

To study the language PL/I with the objective of establishing a standard.

Outline of Program of Work

1. To consider the reports available with a view to eliminating any ambiguous interpretation.

2. To propose improvements which, without modifying the basic philosophy of the language, should ensure the most general implementation, taking also into account specific European requirements.

3. To prepare, as the result of this work, a precise definition of the full language.

4. To consider possible subsets of the language.

5. To establish a suitable machinery for effective co-operation with BEMA and other bodies working on PL/I.

MEMBERSHIP

Chairman:	P. J. Titman (IBM-WTEC)
Vice-Chairman:	A. S. Cormack (NCR)
Members:	J. Dubos (CAE)
	A. Fitzke (ZUSE)
	C. A. R. Hoare (Elliott)
	P. M. Hunt (ICT)
	G. M. Palermo (Olivetti)
	K. W. Ott (ITTE)
	Dr. Schoeniger (UNIVAC)
	F. Sallé (Bull)
	P. Scull (EELM)
	D. Starynkevitch (SEA)

PB1.1.2 CONTROL DATA CORPORATION has made definite commitments that it will have PL/I implemented for its 3000 and 6000 series machines. There is a study group for this purpose in Palo Alto, which will start to work seriously as soon as they find out how much is provided in the IBM implementation.

PB1.1.3 COMPUTER USAGE CORPORATION does not at present have any contract to write PL/I implementation. However, they have a new subsidiary, called Computer Usage Education, which, for a monetary consideration, will be happy to offer a course in PL/I on the premises of any employer who wishes to have employees trained in it.

PB1.1.4 DIGITEK has a contract with Bell Laboratories to provide an implementation for their General Electric machines. It is intended to be ready in May or June of 1966, and will run on the GE635 and output coding for the GE 645.

PB1.1.5 BURROUGHS expects to offer a PL/I implementation with its next set of machines. No work has been started yet pending a firmer definition of the language, but they expect to have a working rudimentary implementation by late 1966 or early 1967, with a version complete enough to deliver to customers available shortly thereafter. They claim that their unique hardware adaptations to software design (built-in stacking, etc.) have kept their ALGOL and COBOL compilers unusually small for their capabilities, and they hope to be able to do the same with PL/I.

PB1.1.6 UNIVAC is planning to release a new product line of compatibles in the modular style in the spring of 1966. Their facility in Blue Bell, Pennsylvania, is working on a PL/I compiler for this line.

PB1.1.7 ALLEN-BABCOCK CORPORATION (ABC in Century City) is implementing a time sharing (remote console) system on a modified IBM 360 model 50, the language to be compatible with PL/I. They are adding a 'LET' verb and only allowing one data type, decimal floating point. This is estimated for completion in the second quarter of 1966. Character and bit-string capabilities are to be added later.

PB1.1.8 RADIO CORPORATION OF AMERICA is not writing an implementation for PL/I at present, but its SPECTRA 70 will have the same instruction repertoire as the IBM 360.

PB1.1.9 The latest word from IBM regarding delivery schedules is that the F Compiler will be ready in April of 1966 (without MACROS or TASKING or SORT) and that the H Compiler will be ready in July of 1967.

PB1.1.10 GENERAL ELECTRIC will provide a PL/I compiler as a part of the automatic programming effort of the MULTICS system operating on the GE 645 computer. This PL/I will be a follow-on compiler of the PL/I used to write portions of the MULTICS system. In addition to these activities, General Electric will strive to standardize PL/I compilers for the 600 line and for General Electric's foreign computer concerns.

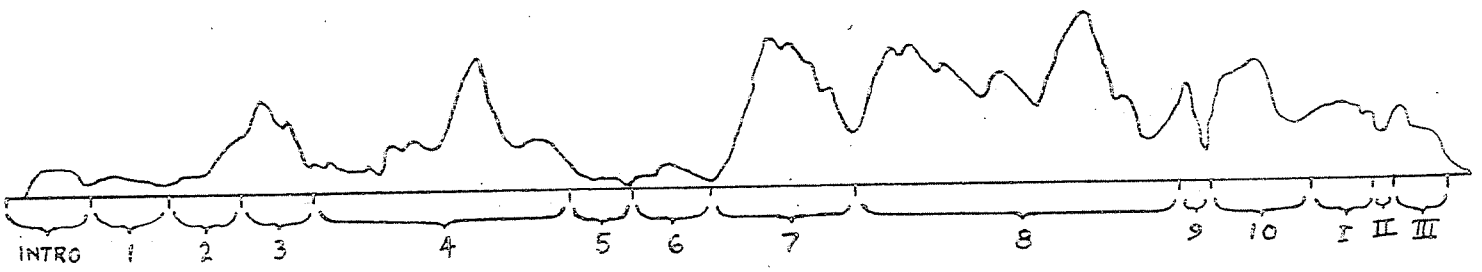
PB1.2 CORRESPONDENCE

PB1.2.1 Comments Concerning PL/I Language Specifications As Published In The IBM Manual (Form C28-6571-1)

California Institute of Technology
Pasadena, California
August 20, 1965

To Whom It May Concern:
Hans Berg, Richard Bigelow, Larry Brown Nobby Clark, Jim Cox, Mark Elson, Doug McIlroy, Roger Mills, John Nash, Ray Larner, Tom Peters, Paul Rogoway, Bruce Rosenblatt, Fred Schneider, Dick Waychoff, Frank Bates

This letter contains the results of a reasonably careful technical review I have just made of the PL/I "-1" manual. I am pleased that the manual has been so much improved; the difference between this version and the last one I saw (May) is almost like the difference between black and white. I congratulate all of the many people who must have put in long hours to achieve this. It took me only 16 hours to read the present manual!! There still are some things which need to be cleaned up, mostly concerned with input-output and mostly easy to fix. I hope that this letter will be of help in polishing up these "last few" details. The following chart, which shows the relative density of "red marks" I made in my copy while reading the manual, may be of interest:



Now for specific points, which I hope I can make sufficiently clear:

1. I think credit should be given to the original language designers: Berg, Cox, McIlroy, Medlock, Radin, Rosenblatt, Sheppard, and Weitzenhoffer. These people contributed so much to the project, it is a shame they are no longer even mentioned. Others like myself who came in much later do not deserve special mention except to say something like "Since the initial design, hundreds of people have contributed valuable ideas to the language," etc.

Precedent for IBM giving such credit in a language manual has already been set, e.g., the FAP manual published by IBM specifically credited Dave Ferguson.

2. p. 12, note 8. The example is bad, since "ר" is not a binary operator.

3. p. 19. The rules imply that condition prefixes have meaning before DECLARE statements (i.e., governing interrupts while the prologue is active), yet the labels which intervene are not. This seems a strange rule; wouldn't it be better to treat DECLARE like FORMAT, i.e., we can GO TO such a labeled statement but it is not executed.

4. p. 24, precision, paragraph 4. Is the precision of "000E5" equal to (1) or (3)? I believe the latter sentence, "If only... 1." should be stricken, since it contradicts the first sentence (which is more consistent and probably more useful).

5. p. 29, initial values, paragraph 2. We read "If a label array element appears in a block, followed by a colon, ..." but nowhere is there any syntax rule stating that this is in fact permissible! We need something inserted, e.g., "A label array element may appear prefixed to a statement in place of a statement label internal to the same block (see "label prefixes" in chapter i), provided the same element does not appear twice." Also make a note of this on page 18. It might be worthwhile to point out that this practice does not apply on END statements, e.g., "END LR(5)" is not permitted. It might also be worthwhile to point out that such a label array name might be qualified, e.g., "A.B.C (2, -5): GO TO SLEEP".

6. p32L (i.e., lefthand column), top. These rules are a little confusing, and for clarity insert "after conversion" after the word "operation" in lines 6 and 14.

7. p32R. The rules for exponentiation would better have rules 1g and 2 like rule 1i, i.e., if x_2 is an integer the operation should be specified as multiplication rather than exponentiation -- it implies greater accuracy. It would be better to define exponentiation first for the case that x_2 is real-valued and fixed of precision (p, 0), giving simple rules in terms of multiplication as in 1d, 1f; then give the rules for all other cases, involving EXP and LOG. By the way, I now have reversed my earlier incorrect feeling that 0**0 should be undefined. I strongly recommend letting 0**0 = 1, not an error condition, since I have found this rule consistent with nearly every algebraic operation. For example, the binomial theorem gives $(x - x)^0 = x^0(-x)^0 = 1$. I feel 0**0 is therefore most useful if regarded as 1, especially in a language which has a macro facility, and this further simplifies the definition of exponentiation (I think).

8. p35L line 6, "that a fixed-point variable of default precision would have been converted to if it had appeared." Isn't this just a clumsy way to say the value is converted to FIXED DECIMAL REAL of default precision -- like we say everywhere else in the manual? The following rule about the null string seems inconsistent, but if it stays we should say, "The null string or a string consisting entirely of blanks is converted..."

9. p35L, coded arith. and decimal: BINARY FLOAT (r) should have p = min(N, r). The "resulting" binary fixed point value is far from clear; I don't know for sure what the result of any of these eight assignment statements is:

DECLARE B(8) BIT (10) VARYING; B(1) = 1.1B; B(2) = -1.1B; B(3) = 1.1E5B; B(4) = -1.1E5B; B(5) = 1.1; B(6) = -1.1; B(7) = 1.1E5; B(8) = -1.1E5;

10. p37, syntax. See my other letters to Mark Elson for a correct syntax. If indeed the priority of "-" has been moved up to equal that of the other prefix operators, I concur, primarily because of examples like "A*-B+C".

11. Page 39, factoring. I observe that conflicting attributes under factoring are no longer allowed. Perhaps an example should be given, e.g., DECLARE A, 2(B, 3C FIXED, 3D, E) FLOAT which is illegal on two counts: the level number of C and D can't override the factored "2", neither can FIXED override the factored "FLOAT". Due to this change in rules, I don't think factored level numbers should be allowed any more, they're now more troublesome to the compiler than is justified by programmer's convenience.

12. p43R, arithmetic data. Add after "numeric picture": "or if they are given no attributes at all." See p. 41, the example at the bottom left column; this explains why TEMP1 and TEMP2 are arithmetic.

13. p47, the label attribute. The word "also" in rule 2 seems incorrect; I think we must sharply distinguish between parameter label variables and others; a parameter label variable should not be allowed to stand for any labels except those which could be passed as argument, lest we GO into the middle of a procedure block.

14. p49, the DIMENSION attribute. The hardest mistakes for me to find in this report are, of course, missing rules which once were present. Here I cannot find anywhere the important rule that the DIMENSION attribute, if present, must come first. Otherwise statements like "DECLARE A FLOAT(10);" are ambiguous. Furthermore there is no longer the prohibition about not factoring the DIMENSION attribute -- a restriction I'm glad to see disappear.

15. Page 51, ENTRY attribute rule 6. An example must be given since the wording isn't sufficiently clear to show the reader what has been said.

16. Page 54, DEFINED attribute. Rules 8 and 9 seem to conflict, and the first part of rule 8 has nothing to do with the next two sentences. Suggested revision: "8. The current generation of the base of AUTOMATIC or CONTROLLED data at each point of ... time of invocation." Drop rule 9 which is in conflict.

17. Page 55, rules for array defining, rule 5 apparently applies only to string arrays, and it is just a bad way of saying rule 6 together with the comments already made under the POSITION option.

18. Page 56, rules for mixed defining. These rules seem to remove most of the restrictions stated above! I cannot follow rule 2c nor can I think what it should be.

19. Page 56, initial attribute. In rule 1, the form should be "[+|-]real-constant [+|-] imaginary-constant"; in rule 5, say "a constant" instead of "an optionally signed constant" -- use the convention of rule 1!

20. Page 57. Why can't the SYMBOL attribute be given for a DEFINED name? Why can't it be given for a parameter? There are probably good reasons, although I can't see why. Surely, for example, a parameter may appear in a data list on input? If not, it is a roundabout way to state the restriction.

21. Page 61. The first paragraph under table 4 communicates nothing to me; how can SEQUENTIAL access apply to direct access devices?

22. Page 62, Assignment of attributes to identifiers. Following point 7, I don't understand the rule about the error which is raised. As stated, BEGIN DECLARE A FIXED; BEGIN DECLARE A ENTRY; BEGIN CALL A END; END; END; would be erroneous (since a containing block has A not declared entry). And is L:BEGIN DECLARE A FIXED; BEGIN ON CONDITION(A) STOP; BEGIN SIGNAL CONDITION (A) END L; erroneous? (I thought condition names were supposed to be external.)

23. Some substitute for IMPLICIT deserves to be in a language like PL/I.

24. Page 63, the rule for "elementary structure element type" is redundant and possibly misleading, so it should be dropped.

25. Page 68, "the arguments in a procedure reference", first paragraph. The whole paragraph is wacky. For instance, a built-in function name is clearly not an expression, and an ENTRY attribute must certainly be required if such a name is ever passed to a procedure.

26. Page 74, synchronizing, last sentence. "After N, the statements O, P, ..., are executed synchronously," etc. -- this sentence seems misleading. I move to strike it from the record.

27. Page 76, purpose of condition prefix, second paragraph. "Enabling of the first five conditions listed above" ... that is some 50 pages above! Strike "first" and "listed above, namely".

28. Page 81. Has the restriction that "no file may appear more than once on the stack of current files at any given time" been removed? It doesn't seem to appear any more. (One reason for having it may be the COUNT function.) Also, how does the stack of current files behave with respect to tasking?

29. Page 83. The first two paragraphs under "list-directed input" have already been stated much more clearly in the preceding discussion, so these paragraphs don't belong any more. The subheading entitled "List-Directed Input Format" forgets to give any rules for the important case when the destination is not arithmetic, e.g., "SUBSTR(XSTRING, 2)" in the example on the left of that page.

30. Page 84, top right. "A data item may span several tabs, but must not span record boundaries." How can this be avoided? Does the output of a long string sometimes cause an error condition simply because it happened to appear at an unfortunate tab position? Does the system make any attempt to help? Also, when the SEGMENT option is being used, it seems that spanning record boundaries is the intent of the programmer; it is virtually unavoidable, and would be the rule rather than the exception! This restriction therefore nullifies the usefulness of the SEGMENT option; yet it appears here and on page 120 as well.

31. Page 85, top left. Formula should be p = MAX(r-s+q, q), lest q be bigger than w!

32. Page 85, coded real floating-point decimal data. The rules imply that "n" on page 90 must be less than or equal to 2. This is a terrible restriction on floating-point exponents which should not appear in PL/I. I suggest the formulas d = p+n-3, s = p+n-2, where n is defined in "floating-point format items". This makes no change in IBM's implementations, and it maintains consistency and virtue.

33. Page 85. List directed output is not generally acceptable as list-directed input, for there are usually spaces before the central sign of a complex data item, violating the rule on page 83 line -3. I suggest squeezing the fields together when the exponent is blanked (cf rule on page 85 line 18) in the real part of a complex output, since the total length in this case is variable anyway, I think, and since tabbing will line things up.

34. Page 86, top left. No mention of string constants appears here. Surely string constants can be used as input to string variables? Also, the form for a constant should refer to the clear definition given on page 83 for list-directed input, instead of "[+|-] constant".

35. Page 86, top left. First it says, "the scalar variable must be unsubscripted" then it turns around and two lines later says it <u>may</u> be subscripted!

36. Page 86. The rules regarding the symbol table seem very silly. The convention that an identifier appearing in a DATA list gets the SYMBOL attribute by default implies there is but one symbol table. Now look at rule 5 on page 86 and the accompanying examples of CARDIN, PARTNO and CARDIN, CARDOUT and see if you can deduce why the funny restrictions have been made at all.

37. Page 88, format-directed, rule 1. I believe sterling data can come out under list-directed output using a decimal numeric field (page 85L, bottom). Hence also for DATA-directed output.

38. Page 88, examples. More appropriate to write "(A, X(2), 20F(2))" for the format in example 2, since the first part of that example is abundantly meaningful yet I doubt if the last part was.

39. Page 90, top left. In option 3 the scale factor should <u>divide</u> not multiply --- on either input or output, but not both, I forget which it should be, as in the FORTRAN convention. The operations of input and output should be inverse! Also, at the end of the paragraph we read "If d is omitted, only the integer portion of the number is considered." First, it is not shown that d <u>can</u> be omitted on option 3; second, is only the integer portion considered <u>before</u> or <u>after</u> scaling?

40. Page 89-90. What zero-suppression, if any, is done on F format? How is the value zero output under, say, F(5, 1)?

41. Page 91. Under A format, rule 1 for bit strings conflicts with rule 4 for character strings. This is an undesirable inconsistency, particularly since the same letter A is used in both cases.

42. Page 91. What does A(w) mean for numeric (picture) fields on input? Example: picture is '$$$.99', data card contains "+1E0"; do we input '$1.00'? If the data card contains "$1.00" I presume this is an error (p forward should have been used), since it is a character string

that cannot be converted to a number for transmission to the picture. I do not object to these rules, but I think they ought to be pointed out to commercial users. (Added later: Perhaps I'm wrong here, but what about A(w)., for coded numeric input, and what about P for coded numeric input?

43. Page 92. Why give the rules for "internal picture format items" and "internal character-string format items" when these rules have been given on the previous page. I miss the distinction implied by the word "internal".

44. Page 94. Key option, rule 2. The last clause "if the key does not exist, the record is added to the file," seems to almost invalidate the NEWKEY option, and it also seems to contradict the sentence at the top right of page 94.

45. Page 95. When is the expression of the REGION option evaluated? Is it like the KEY and NEWKEY options in this regard, i.e., the expressions are evaluated later in some other block during the processing? Does a single READ or WRITE statement always refer to just one region, even if CROSS applies?

46. Page 96. Other ways in which sequencing is altered are (a) during prologues; (b) during certain ALLOCATE statements; (c) an input or output process causes the expression from a KEY or NEWKEY option to be evaluated; (d) an array defined in another array.

47. Page 97. Isn't STRING a pseudo-variable?

48. Page 98, bottom left. The code is not necessarily illegal if B is a parameter.

49. Page 99. In the structure assignment "S = T" may the structures contain label variables in corresponding positions?

50. Page 99. No rule is given explaining structure assignment that is not BY NAME!

51. Page 99. Rule 1a should only apply to the first case of option 4, not the second (probably); e.g., L1(I, *) = L2(J, *) where I, J are abnormal.

52. Page 100. Rule 4a is inconsistent with other rules; for example the structure assignment A(I+J) = 0 not by name evaluates I+J several times, as does the statement A(I+J, *) = A(1, 1).

53. Page 100, after rule 4f. The next paragraph should be rule number 5. The phrase "or the label variable" should I think be stricken. For example, what is the answer in the following program:

```
MYSTERY: PROCEDURE OPTIONS (MAIN) RECURSIVE;
    DECLARE (I INITIAL (0), (L, M) LABEL) STATIC, J;
    J = I;
    IF I=0 THEN DO; I=1; L=PRINT; CALL MYSTERY END;
           ELSE DO; M=L; GO TO M; PRINT:WRITE DATA (J) END; END;
```

The answer is either "J=1" or "J=0" -- I strongly believe it should be "J=0" but the stated rule seems to say otherwise.

54. Page 102. The list of possible parameters does not include event-name arrays, task-name arrays, or structures which contain labels, events, or tasks.

55. I have a general question about order of evaluation of expressions appearing in options, and throughout input-output statements -- I never

expect this to be answered satisfactorily in this PL/I manual, it is much too difficult a problem, but I hope some student makes it his Ph. D. thesis.

56. Page 102, call statement, syntax rule 3. No, a blank is not necessary; e.g., "CALL PRINT(A, B) TASK (T2)PRIORITY(-2);" needs only one blank. Similarly, syntax rule 2 under CLOSE is wrong on page 103.

57. Page 103, call statement, general rule 2. Here "entry names" and "built-in function names" are regarded as distinct types; then there is no rule that says built-in function names can be arguments in any case. This is like saying "One must always use apples or oranges except in case X when peaches cannot be used."

58. Page 103 and elsewhere. In the presence of factoring in its many appearances, are factored expressions evaluated more than once? Consider as one example CLOSE(F, G) CALL A; does A have to check the label for both files? This is impossible since the stack of current files always has but a single file at the top, so GET and PUT could not be used for multiple files. Therefore, A must be called twice. Another example: DECLARE(X, Y) CHARACTER(F(3)) INITIAL CALL G(4); how many times are the procedures F, G invoked here? I would guess perhaps G should be invoked only once!

59. Page 104. The examples of CLOSE are ill-chosen since they don't even attempt to show factoring or the IDENT option or anything interesting.

60. Page 105. The task option of DISPLAY is not explained. Also, semicolons have been left out of the "general format" (both options) by mistake.

61. Page 107. As I understand tasking, encountering the END statement which terminates a task causes something to happen to an event variable. Neither the termination of a task, nor the changing of event variables, nor the waiting for attached tasks to stop, seem to be mentioned under the END statement or elsewhere.

62. Page 108, FETCH, rule 3. What are task identifiers -- hasn't that term become obsolete? And isn't this rule stated better on page 139?

63. Page 108, FORMAT. The syntax calls for one label followed by zero or more colons! It should be, as usual, "label: [label:]... FORMAT format-list;". Same deal on PROCEDURE statement, page 118.

64. Page 110, GO TO, rule 4. This rule must be incorrectly stated, since there is no point for having a restriction that can always be easily circumvented. The original rule was designed to ensure proper maintenance of the stack of current files. The rule should therefore be, e.g., "A GO TO may not terminate any procedure invoked with the CALL option in an input-output statement." Even GO's out of ON units must be restricted thus; or else the GO TO process should check for properly handling the stack of current files and the restriction should go out altogether!

65. Page 110, GO TO, rule 6. Obviously this rule is true, but you want to say more strongly that a GO TO may not terminate (or GO out of) a procedure invoked with any task option.

66. Page 111. Does GROUP interact in any way with PAGE?

67. Page 112. Better to drop syntax rule 2, which is confusing, and instead add the parenthetical remark "(The IF statement is not itself terminated by an additional semicolon.)" Also, the examples and discussion are unfortunate, and they have led many people to needlessly criticize PL/I. In fact, rather than use null ELSE's, it is always better, and conceptually satisfactory, to use groups, e.g.

```
A: IF X > Y THEN DO;
            IF Z = W THEN IF W < P THEN Y = 1; ELSE X = 4; END;
             ELSE X = 4;
J: Z = 5;
```

68. Page 113, ON. Important: PL/I can lead to several situations when an expression appearing in one block must be evaluated within a dynamically encompassing block. The question is, which generation of data is used then? For example, I ask what is the output of the following program -- I am almost certain Doug McIlroy and John Nash will give different answers!

```
TEST: PROCEDURE OPTIONS (MAIN) RECURSIVE;
      DECLARE I STATIC INITIAL (1), (A, B) FIXED, F FILE
          INDEXED DIRECT KEYLENGTH (1) ENVIRONMENT(CREATE);
A, B = I; IF I = 1 THEN DO; I=2;
                            ON CONDITION(S) A=3;
                            K:PROCEDURE;B=3;RETURN('*');END;
                            TRICK:WRITE FILE (F) CALL TEST
                                  NEWKEY(K) CROSS HOLD;
                            WRITE DATA(A, B) END:
          ELSE DO; SIGNAL CONDITION(S);
                   PUT ('GEE WHIZ') (SPACE, A) END; END TEST;
```

The point here is that the statement labelled TRICK calls TEST recursively; now there are two copies of the AUTOMATIC variables A and B. The recursive call signals an ON condition that was enabled in the outer call, and also calls for the evaluation of a KEY expression -- which values of A, B are thereby used? The output should be "A=1, B=1" or "A=3, B=3" or some such thing. Note: I tried to include the expressions in array defining, or the LIKE attribute, or ALLOCATE, into this example, but couldn't think of any problem situations of this kind.

69. Page 115. Since apparently the ACTIVITY option can't be explained in a meaningful machine-independent way (I for one have no idea what ACTIVITY(1) or ACTIVITY(100) means) it belongs as part of the ENVIRONMENT option which is already present on OPEN.

70. Page 116. On the IDENT option for both OPEN and CLOSE a data list appears -- the report should say it is an output data list.

71. Page 116. Is there any relation between a page and a group, a line and a record? Also, Rule 1: When the PAGE statement is given and a skip is thereby caused, are footings put out on the previous page? Also Rule 6: Does the SIZE option count all blank lines including margins (or is there no way to specify vertical margins)?

72. Page 118. The example at top left is very bad since the format specification "X(6)" will be ignored! This will probably be a common programming error, but people will learn.

73. Page 118, MAIN is called an attribute on page 71, now an option on page 118. By the way, it is a nuisance writing "OPTIONS(MAIN)" as my examples in points 53, 68 show; and merely writing it as we write RECURSIVE would be preferable. MAIN is not implementation-defined, it is defined for all PL/I implementations on p. 71.

74. (Useless comment deleted.)

75. Page 124, SEGMENT. Wouldn't example 2 be better if it were, say, SEGMENT(';');.

76. Page 126. The SORT still needs pre- and post-editing capability.

77. Page 128, just before "WRITE": "...until the attached task is completed, or until the EVENT pseudo-variable is referred to in the other task in an appropriate manner."

78. Page 129, top right, line 10. "Record boundary crossing due to LIST, DATA or SEGMENT does not require the CROSS option." How can this be -- apparently list-directed and data-directed output normally create more than one record, contrary to rule 3? And SEGMENT implies CROSS as stated later -- or does this use of the word SEGMENT refer to a SEGMENT statement occurring during procedure-directed transmission (which surely ought to have CROSS).

79. Page 130, end. The example is not in a form suitable for data-directed input as claimed, since no semicolon appears as required on page 86.

80. Page 134. Paragraph 6 says a parameter can't be STATIC or AUTOMATIC; page 72R paragraph 2 says "every variable must have a storage class." Therefore every parameter which is a variable is CONTROLLED. But no; we read on p. 136 that somehow a parameter may have no storage class. Some clarification should be made.

81. Page 134R line 8. The word "may" is dreadfully ambiguous here.

82. Page 135. In the example programs, if A and B are internal procedures, is the ENTRY attribute really necessary? If so, say so since some programmer will expect the compiler to know a "simple" thing like the information in that ENTRY attribute.

83. Page 135. Why make a dummy argument for a label array?? (Cf. point number 53 above.)

84. Page 137. The example program sets PP=Q before PP is allocated.

85. Page 140. In the description of FIXED, shouldn't the second argument be the total number of digits, and the third the number of digits to the right of the point?

86. Page 145. May the operator in SCAN be 'CAT' or '>' or 'GT'? I can't think of any useful application except SUM, PROD, ALL, ANY which we already have in the language. Who wants SCAN anyway?

87. Page 145. Is ONCODE always set when ERROR is raised? Is ONCODE set at any other times?

88. Page 145. In the description of POINT, the meaning of logical record is not clear. Neither is the POSITION statement, page 117, when the SEGMENT option is in effect.

89. Page 146. The ROUND function belongs among FLOOR, CEIL, TRUNC, etc., not on this page.

90. Page 147. Everyone I know including myself thinks it is a mistake to print "DB" for non-negative fields. The rule in COBOL is to treat DB exactly the same as CR - reason is that when computing someone's bill, the amount due was negative. PL/I cannot negate common data-processing practice in this way and expect to win friends.

91. Page 152. I don't get any sense out of the ACCESS condition. Also under IDENT, strike the words "return from the ON unit. Processing will" in order to be consistent with language used elsewhere. Also in the system action for TRANSMIT, reference is made to some ON unit which is not present during standard system action!

92. Page 152. The CHECK conditions are said not to arise during prologues, but I wonder how or why the compiler suppresses them, say, while evaluating a programmer-defined function as part of a prologue.

93. Page 153. The statement Y = F(X)+F(X) may set X to two different values, with considerable amounts of computation going on in between, but we read that the CHECK condition is raised only once. Really? Which time?

Well, that concludes my list. You see, I couldn't even get up to 100 points this time -- in a language as complex as PL/I, that is remarkable, considering what an awful nit-picker I am.

Perhaps I have been off the track in several of the comments above; but even so, other readers are going to wonder about the same things, and the manual ought to be changed somehow to clarify the situation.

Actually I have another list besides the above, those things which I called "typographical errors" although they are not all so simple. In fact many of the corrections in the following list are not at all obvious since they require a good knowledge of the full language; and I feel even to point out the obvious ones will be helpful since the manual is reasonably close to its final form. You will be wise to check me on each of these corrections. (See the table on the next page.)

Further corrections are necessary also to most of the examples of READ and WRITE statements in the manual. I wasn't sure where the error lies -- either the syntax rules for READ and WRITE should state that the options may optionally be separated by commas, or else all the commas should be deleted in some 20 places. Here are the places where commas would need to be deleted:

(The list of places where commas would need to be deleted, and the list of typographical errors are omitted. We felt that these were not of general interest. Editor)

Now I am ready to retire. I wish I could come to IBM to discuss each of these matters in person, but since that is impossible I sincerely hope this letter adequately conveys my intentions, and that you will spend as much time studying this letter as you would spend with me if I were there to work with you.

Regards,
Don Knuth

PB1.2.2 PL/I and Its Future

Lockheed Missiles & Space Company
52-40, Building 201
3251 Hanover Street
Palo Alto, California 94304

Thank you for the invitations to contribute to and to receive the PL/I Bulletin; I should like to accept both. A quick outline of my thoughts on PL/I follows:

PL/I represents, I think, a fundamentally bad strategy. Its language, far too rich for most applications, is at the same time far from universal; the resulting compromise is a language not quite right for anything. This is the inevitable consequence of its designers' error of aiming toward a universal language of conventional implementation rather than toward an open-ended processor capable of translating a growing variety of languages. This is much too involved an argument to present with any thoroughness here; I hope soon to publish a paper that will do so. As to PL/I's future: the weight of IBM is enough to ensure that PL/I will be widely implemented and strongly urged upon us; how many programmers will voluntarily abandon FORTRAN and COBOL to use it is a very different question. If it is to find widespread use, I suspect it will come about only through a strong IBM sales-pitch to non-programming middle and upper management on the economic virtues of a single-language shop, followed by much arm-twisting of working programmers. In one way I welcome PL/I: I think it will provide a massive test of my hypothesis on what's wrong with closed processors. In the hope that its failure will speed the coming of the age of mammals, I welcome this last and greatest of dinosaurs.

Yours sincerely,
Mark Halpern

PB1.2.3 PL/I and Its Future

RCA-EDP
Cherry Hill - Building 204-2
Camden, New Jersey 08101

Thank you for your kind invitation to comment on PL/I for the PL/I Bulletin, your offer to place me on the mailing list (which I accept), and your suggestion that my crystal ball, unlike everyone else's, is capable of predicting the future of PL/I (it isn't).

Rather than prognositcate, I should prefer to ask, or re-ask, some questions that have occurred to me. I have been given answers to some of them, but the answers have uniformly been based on experience with PL/I, and hence require re-evaluation each time the language changes, which is, apparently, monthly.

Perhaps the single most fundamental question that can be raised is: What is the purpose of PL/I? The trite answers ('All things to all men') having been dismissed, I re-ask the question, and without prejudice. Is it a meaningful concept to have a single language which purports to cover all of computation?

Lest I be accused of biassing the answer by my phrasing, let me ask some ancillary questions. Is PL/I an improvement over established languages for some classes of problems? Does PL/I cover a class of problems for which there are no extant languages? What price has to be paid for the generality of PL/I, and is the price acceptable to those who have a choice (as contrasted with those who, for political or managerial reasons, may not)?

There is, of course, the question: What is PL/I? This, although a valid question, appears to me to be an unfair one, since it is acknowledged by the proponents of the language that it is still under development, and that changes can be expected, and are, indeed, required. On the other hand, having conceded the unfairness of the ontological query, I consider the pressure being applied for the prompt acceptance of PL/I to be equally unfair. Either it is to be judged on its present merits (in which case, I suspect that it would fail), or it should be judged on its probable outcome (in which case its present merits are less important, and hence should not be pressed). The programming world does not need another Jenny Haniver.

In summary, I would wish to observe that there is evidence of a great deal of careful thinking in the language PL/I, and equal evidence of its requiring much more of the same care. I look forward to this Bulletin serving as a forum for those interested in the language, and hope that it will have some effect on the direction that development in the language takes.

Sincerely,
Peter Zilahy Ingerman

PB1.2.4 PL/I and Its Future

Moore School of Electrical Engineering
University of Pennsylvania
Philadelphia 19104

PL/I is certainly a new and flexible language, and should be a commercial success, what with its backers. As would be expected, most of the ALGOL ideas have finally been adopted and extended. It is unfortunate that the present giant effort could not have been used in support ofALGOL beginning in 1957.

However, it appears to a non-participant that the PL/I project is still approaching the past's problems with a technology out of the past. Although

supposedly machine independent, it could not break with the machine structures into which it is embedded. Therefore, it will probably fail to lift the act of computer programming out of the area of detail and into the area of analysis, where it needs to be. One certainly must wish its supporters success in the limited venture to which it appears to apply. The basic idea of "growing machine", extendible syntax, and rationalization of semantics appears (to an observer who has neither participated in the PL/I design, used it, nor taught its use) to be still unmet.

The PL/I designers deserve great commendation for going as far as they have, particularly with regard to facing up to the great programming shibboleth, 'efficiency', which always rears its head, like Prohibition in the 1920's. (Nobody believes it's really possible, but everyone is afraid to say so.) Perhaps the introduction of PL/I may lead to an analysis of the efficiency of the overall system, not just that part lying within the computer.

There still need to be computer systems of usage reserved for more intelligent communication between man and machine. I hope the users of PL/I, as well as its designers, will evaluate their efforts critically in that direction, and not stop at efforts to improve man-computer communication.

I want to thank you for asking me to comment. That is a compliment in itself. If you have any questions or suggestions, I'll be pleased to give them. I will be pleased to receive PL/I bulletins, and wish you success in your venture.

Sincerely yours,
John W. Carr III

PB1.2.5 PL/I and Its Future

Computer Analysts & Programmers Ltd.
62-63 Queen Street
London E. C. 4.

Thank you very much for your letter of the 10th December informing me of the new PL/I Bulletin. I am most interested to hear of this and hope to make a joint contribution with my colleague Mr. Fraser Duncan.

My initial impression is somewhat parochial. I greatly fear that this country will 'sit on the fence' about this language too long to make an effective contribution to its development, or even to produce competitive compilers in the near future.

It seems to me that although one might criticise the details of the language it renders doubtful the validity of further extensions to ALGOL and FORTRAN.

Yours sincerely,
A. d'Agapeyeff

PB1.2.6 PL/I and Its Future

C.E.I.R., Inc.
Applied Research & Management
Sciences Division
1200 Jefferson Davis Highway
Arlington, Virginia

I am very pleased to hear of your intent to publish a PL/I Bulletin and would be most interested in receiving copies.

When PL/I was first announced, I frankly felt no small amount of apprehension. This condition had little to do with the content of the language at that time or with any of its names. My worries were concerned with the need for any new language and the appropriateness of a mere six headed Committee to specify it. Furthermore, I was concerned with the moral responsibility to the entire community and particularly whether this responsibility would be understood by IBM.

Over the past months the language has undergone significant change. If these changes continue for the general good, that is, if there is a serious effort to create a superior product, then the question of need becomes academic. I do believe, however, that with the tools available in the current state of the art, any new language must make substantial contributions over what exists if this need is to be valid.

The other misgiving that I had initially seems to be waning. I always worry when a small group within a closed environment speaks for the entire technical community. I could never subscribe to the notion that what is good for SHARE is good for the world. While this indeed may be the case, I always feel more comfortable operating in the open. It is healthy for other activities and disciplines to be involved in an activity which has such far reaching implications. I understand that the Business Equipment Manufacturers Association and the European Computer Manufacturers Association share this view. The publication of a PL/I Bulletin will offer all interested and capable individuals an opportunity to participate. Although this might tend to delay a consensus, I believe we are obligated to do so.

My final concern appears to be growing. Should a programming language be marketed as a product? This does not imply that it should not be made available, but what worries me is the gaining of an economic advantage by

marketing PL/I while at the same time offering to place PL/I in the public domain. Is it really a good product or are we all being told to eat spinach because it is good for the farmers?

This last question will certainly be answered this year when installations will have the opportunity to use the language and its processors. Currently my feeling is that the language has far greater appeal to the Fortran types than to any other and that installations that do 50% scientific and 50% commercial programming really do not exist. If this language is really a significant improvement over existing tools, its success to a great extent will depend upon its presentation. ALGOL, for example, failed because it was never clearly and simply presented. Instead of being all things to all people simultaneously, PL/I should be shown to be most things to one person at one time. Finally, I think it would be prudent to make other than naive comments concerning PL/I's relationship to the existing major languages.

It is not sufficient merely to combine ideas. We must discriminate and perceive where the synthesis leads. From this will come enlightenment and, ultimately, achievement.

Sincerely,
Howard Bromberg

PB1.3 WORKING PAPERS

PB1.3.1 PL/I Publication Guidelines

F. W. Schneider

From past experience each of us knows of the difficulties inherent in trying to read another person's code, and yet this is just what we are asking people to do in the "Programs" section of the PL/I Bulletin. In order to alleviate some of the problems involved in the transmission of procedures, WG4 felt it necessary to set up some guidelines and requirements concerning the format of programs to be published. The principle aim of these rules is to make it easier to grasp the overall structure and and flow of a code presented on a printed page.

The rules decided upon were:

1. Procedures must be written in PL/I and must stand alone as a unit of information. If references to outside procedures are made this must be done in such a way as to allow anyone reading the code to understand it without having to read the outside procedures. Comments and complete declarations are principle means for describing the functions performed by such external procedures. Such documentation can also be provided by using a previously published program and referring to its number in the PL/I Bulletin.

2. The use of characters which are neither in the PL/I syntactic character set nor lower case letters is limited to comments, unless required by the procedure. It should be taken into account that upper and lower case letters are indistinguishable. No two names may differ only by the case in which they are written, rather once a particular form is chosen it should be used constantly within the procedure. Some procedures, such as editing and typewriter handling routines, will require the use of non-standard characters. In this case the non-standard characters may be used only in the comments or in strings, and their non-standard nature should be pointed out in comments. It has been suggested that other non-standard characters

which are homomorphic to the syntactic character set should be allowed. This includes subscripts, superscripts, and the standard mathematical relational operators. Our feeling in this matter was that the programs should be in such a form that they could be easily rendered into a machine-readable form by unskilled personnel, such as keypunchers and terminal operators.

3. Privileged words (those recognized as meaningful in context by a compiler) should be boldface, E.g.: READ: READ DATA(DATA); GO TO READ;. This is required to make reading of the procedures quicker and simpler. No confusion can arise concerning the usage of a name which might be either a variable or a statement name. (Bold face is indicated by underline in typescript.)

4. Any program characteristics which are implementation dependent must be explicitly defined. It is "nice", reasonable, and often necessary to write procedures which take advantage of some feature of the compiler, the data, or the processor. Any procedural characteristics making use of such a feature, however, thereby binds itself to that particular hardware or software. Any such ties must be pointed out in the procedure. The characteristics involved must be explicitly stated and illustrated in order that the procedure be made meaningful to all readers.

While the above rules are regarded as the minimum to make the procedure potentially useful to any reader, it is obvious that some more detailed instructions as to the format of a procedure are required. The following, then, are some recommendations to the potential contributor. This might also serve as a basis for a documentation procedure for any coding done in PL/I.

In PL/I a label appears at the beginning of a statement. Unlike some languages, however, PL/I allows a statement to appear anywhere in an input

stream. This makes it difficult to locate labels when reading the program. Therefore it is recommended that all labels (with their colon) be placed at the left margin, followed by the associated statement, indented as needed.

More important to the understanding of a procedure than labels and GO TOs is the block structuring. A block is contained between a PROCEDURE (or PROC), BEGIN, or DO statement and the corresponding END statement. In order to give a graphic idea of the structure of the procedure it is recommended that END statements which are not on the same line as the PROCEDURE, BEGIN, or DO statement of the outermost block which they end should be directly below this PROCEDURE, BEGIN, or DO statement and that all intervening lines should be indented a standard number of spaces (preferably 5) from the beginning and end of the block. If an entire IF statement will not fit on one line then the continuation on the next line should be indented the same standard number of spaces from the IF. The ELSE, if any, should be directly below the IF and follows the same rule on continuation. These recommendations on indenting will obviously have to be relaxed in the case of a deeply nested program.

One final and seemingly trivial point concerns spacing. An evenly and consistently spaced code is much more pleasing and easier to read than one in which some lines are all run together and some are spread way out. It is recommended that spaces be inserted after, but not before, commas, colons, and semicolons except within parentheses. Parenthesized items other than parameter lists and subscripts should be surrounded by blanks. Judicious insertion of spaces can also be used to make a complicated expression easier to read by bracketing the operators with blanks, the higher the binding strength, the fewer the blanks.

While the working group will naturally not refuse to publish worthwhile programs which do not conform to our rules, we may deem it necessary to rewrite them in our format. Finding the time to do such a rewrite could introduce a regrettable delay in getting the procedure out to the readers.

Page 23 Missing

```
                A(i, k) = A(i, k) - DOT(A(i, *),A(*, k), 1, k-1);
                IF ABS(A(i, k))>temp THEN
                DO ; temp = ABS(A(i, k)); imax = i; END;
            END;
            pivot(k) = imax;
            /* We have found that A(imax, k) is the largest pivot in column
            k. Now we interchange rows k and imax */
interchange:IF imax = k THEN GO TO eliminate;
            DO j = i TO n; temp = A(k, j); A(k, j) = A(imax, j);
                A(imax, j) = temp;
            END;
            temp = b(k); b(k) = b(imax); b(imax) = temp;
eliminate:  IF A(k, k) = 0 THEN GO TO singular;
            quot = 1/A(k, k);
            DO i= k+1 TO n; A(i, k) = quot*A(i, k); END;
            DO j= k+1 TO n; A(k, j) = A(k, j)- DOT(A(k,*), A(*,j), 1,
            k-1); END;
            b(k) = b(k) - DOT(A(k, *), b, 1, k-1);
        END TRIANG;
        GO TO subst;
SOLVE: ENTRY(A, b, y, pivot, DOT);
        /* SOLVE assumes the matrix A has already been transformed by a
        previous call of CROUT, and that pivot has been set accordingly.
        SOLVE gives the same solution to Ay = b as CROUT would have given
        for the original matrix A. However SOLVE is faster, because it
        does not repeat the triangularization of A. */
        n = HBOUND(b, 1);
        DO k = 1 TO n; temp = b(pivot(k)); b(pivot(k)) = b(k); b(k) = temp;
            b(k) = b(k) - DOT(a(k, *), b, 1, k-1;
        END;
subst: /* The triangular decomposition is now finished, and we do the back
        substitution. */
        DO k = n TO 1 BY -1; y(k) = (b(k) - DOT(A(k,*), y, k+1, n))/A(k,k);
END CROUT;
```

PB1.5 PITFALLS & PRAGMATISMS

PB1.5.1 Array Arithmetic

F. W. Schneider

A major pitfall to the beginning scientific user exists in PL/I, and from the looks of things it's going to stay there. Beware of the so-called "array arithmetic". It's nothing but a shorthand notation for an iteration.

Example:

```
DECLARE a(2, 2) float initial (4, 5, 6, 7);
DECLARE b(2, 2) float;
```

At this point if you say: b = a/a(1, 1);
the value of the expression a/a(1, 1) is different than if you had said:
a = a/a(1, 1);

Fine, you say, now that I know about it I simply won't put the same variable on the left of the equals as on the right in an array arithmetic statement. Wrong again; you have no way of knowing whether or not two of the arguments to a procedure are going to be the same. For example:

```
NORM: PROCEDURE (A, B, X); DECLARE (A(*, *), B(*, *), X) FLOAT;
    A = B/X;
END NORM;
```

works just fine until you say: CALL NORM(A, A, A(1, 1));
and then you're right back where I started.

PB1.5.2 Fixed Point Arithmetic

Bill Rosenthal

Not only does truncation occur with trailing digits, but in PL/I truncation also occurs with the most significant digits.

If we add 25 + 1/3 in fixed point arithmetic we may obtain the result 5.333333333. The leading digit 2 is truncated.

Assume 10 digit counters

0	0	0	0	0	0	0	0	0	1	10 digits
0	0	0	0	0	0	0	0	0	3	10 digits
0	0	0	0	0	0	0	0	2	5	10 digits

N = the length of the largest number in the implementation.
m = the total number of positions in the result.
n = the scale factor of the result.
p = the total number of positions in operand one.
q = the scale factor of operand one.
r = the total number of positions in operand two.
s = the scale factor of operand two.

for division

m = N m = 10

n = N-p+q-s n = 10-1+0-0=9

Therefore 1/3 = 0.333333333

now for addition 25 + 0.333333333

m = min(N, max(p-q, r-s) + max(q, s) + 1) = 10

n = max(q, s) = max(0, 9) = 9

∴ 5.333333333

PB1.8 REFERENCES

PB1.8.1 "NPL Technical Report", Form No. 320-0908, (December 1964) International Business Machines Corporation.

PB1.8.2 "Report of the SHARE Advanced Language Development Committee", Report No. I, (March 1, 1964), International Business Machines Corporation.

PB1.8.3 "Report of the SHARE Advanced Language Development Committee", Report No. II, (June 24, 1964), International Business Machines Corporation.

PB1.8.4 George Radin and H. Paul Rogaway, "NPL, Highlights of a New Programming Language", Communications of the ACM, (January 1965), Volume 8, No. 1, p. 9.

PB1.8.5 "PL/I: Language Specifications, IBM Operating System 360", Form No. C28-6571-1, (July 1965), International Business Machines Corporation.

PB1.9 CIRCULATION LIST

R.ANDERSON
1212 CENTINELA AVE.
LOS ANGELES, CALIF. 90025

WAYNE D. BARTLETT
CONTROL DATA CORP.
11428 ROCKVILLE PIKE
ROCKVILLE, MARYLAND

FRANK J. BAUER
10572 ASHTON AVENUE
LOS ANGELES CALIF. 90024

LEE A. BEGHTOL
BECKMAN INSTRUMENTS - SYSTEMS DIV.
2400 HARBOR BLVD.
FULLERTON CAL.

N.D. BREWER
SDS
2526 BROADWAY
SANTA MONICA, CAL.

HARRY N. BROOMALL
BECKMAN INSTRUMENTS
2400 HARBOR BLVD.
FULLERTON, CAL.

D.E.W. BUCHER
COMPUTER SCIENCES CORP.
650 NORTH SEPULVEDA BLVD.
EL SEGUNDO, CAL. 90245

R.G. CANNING
134 ESCONDIDO AVE.
VISTA, CAL. 92083

JOHN W. CARR III
MOORE SCHOOL OF ELECT. ENG.
UNIVERSITY OF PENNSYLVANIA
PHILADELPHIA, PENN. 19104

JERRY CARRICO
UNION BANK
1024 SOUTH HOPE STREET
LOS ANGELES 15, CAL.

G. MILLER CLARKE
105 WEATHER VANE DRIVE
CHERRY HILL, NEW JERSEY 08034

VIRGINIA COHEN
SDC
2500 COLORADO AVE.
SANTA MONICA, CAL. 90406

TECHNICAL LIBRARY
COMPUTER SCIENCES CORP.
650 N. SEPULVEDA BLVD.
EL SEGUNDO, CAL. 90245

2 TECHNICAL LIBRARY
COMPUTER USAGE EDUCATION, INC.
51 MADISON AVENUE
NEW YORK, NEW YORK 10010

CORPORATE TECHNICAL LIBRARY
C-E-I-R, INC.
1200 JEFFERSON DAVIS HIGHWAY
ARLINGTON, VIRGINIA 22202

J.L. COX
IBM U.K. LABORATORIES LTD.
HURSLEY HOUSE, HURSLEY PARK
WINCHESTER,HAMPSHIRE, ENGLAND

LEO E. DAVIS, JR.
IBM
7220 WISCONSIN AVENUE
BETHESDA, MARYLAND 20014

C. DES COURTIS
SACS
35, BOULEVARD BRUNE
PARIS 14EME (SEINE) FRANCE

PHIL DORN
C A I
555 MADISON AVENUE
NEW YORK 22, NEW YORK

GEORGE A. EDES
SECURITY FIRST NAT. BANK
514 SOUTH SPRING STREET
LOS ANGELES, CAL.

PROF.INC. PAOLO ERCOLI
C.N.R.
PIAZZALE SCIENZE 7
ROMA ITALY

PROFESSOR S. GILL
DEPT. OF ELECT. ENGNR.
UNIVERSITY OF LONDON
LONDON S.W.7 ENGLAND

F. GENUYS
IBM WORLD TRADE EUROPE
8-10, CITE DU RETIRO
PARIS 8EME(SEINE) FRANCE

JOHN A. GOSDEN
AUERBACH CORPORATION
1815 N. FORT MYER DRIVE
ARLINGTON, VIRGINIA 22209

A.G. GRACE, JR.
RCA EDP
CHERRY HILL BLDG. 204-2
CAMDEN, NEW JERSEY 08101

IRWIN GREENWALD
RAND
1700 MAIN STREET
SANTA MONICA, CAL.

DAN HAGGERTY
SDC
2500 COLORADO BLVD.
SANTA MONICA, CAL. 90406

MARK HALPERN
LOCKHEED MISSILES + SPACE CO.
52-40, BLDG. 201
PALO ALTO, CAL. 94304

JAMES HARKINS
G.E. COMPUTER EQUIPMENT DEPT. B113
13430 BLACK CANYON HIGHWAY
PHOENIX, ARIZONA

EMANUEL HAYES
SDC
2500 COLORADO AVENUE
SANTA MONICA, CAL. 90406

D. HEKIMI
ECMA
RUE DU RHONE 114
GENEVE, SWITZERLAND

20 MR. VICO HENRIQUES
BEMA
235 E. 42ND STREET
NEW YORK, N.Y. 10017

R.A. HOLT
2515 R. STREET S. E.
WASHINGTON, D.C. 20020

MR. PETER ZILAHY INGERMAN
RCA - EDP
CHERRY HILL - BLDG. 204-2
CAMDEN, NEW JERSEY 08101

IBM CORP., ASDD LIBRARY
2651 STRANG BLVD.
YORKTOWN HEIGHTS, N. Y. 10598

WILLIAM H. ISAACS
MICHAEL REESE HOSPITAL
29TH STREET AND ELLIS AVENUE
CHICAGO, ILLINOIS 60616

G.D. JOHNSON
UCLA COMPUTING FACILITY
BOELTER HALL RM. 3532B
LOS ANGELES, CAL. 90024

ROBERT E. JOHNSON
NATIONAL CASH REGISTER
2815 WEST EL SEGUNDO BLVD.
HAWTHORNE, CAL.

R.B. KELLER
IBM CORP.
525 FLOWER STREET
LOS ANGELES, CAL. 90017

DENNIS E. KELLEY
21916 S. IBEX AVE.
HAWAIIAN GARDENS, CAL. 90701

NEAL M. KENDALL
NATIONAL CASH REGISTER
2015 WEST EL SEGUNDO BLVD.
HAWTHORNE, CAL.

H. C. KERPELMAN
RCA EDP
CHERRY HILL BLDG. 204-2
CAMDEN, NEW JERSEY 08101

PROF. DON KNUTH
SLOAN LABORATORY
CAL TECH
PASADENA, CAL.

GEORGE KREGLOW
AUTONETICS 1074/N
7312 SOUTH JEFFERSON STREET
ANAHEIM, CAL.

T. D. C. KUCH
ADR, INC.
1815 N. FORT MYER DRIVE
ARLINGTON, VIRGINIA 22209

LEN LONGO
14521 MORAN AVENUE
WESTMINISTER, CAL. 92683

M. D. MC ILROY
BELL TELEPHONE LABORATORIES
MURRAY HILL, NEW JERSEY 07971

ED (MADMAN) MANDERFIELD
12900 LAKEWOOD APT. 2
DOWNEY, CAL.

DR. NORMAN M. MARTIN
4423 CRESTWAY
AUSTIN, TEXAS

PROF. ROBERT J. MEYER
DEPT. OF MNGMNT., COLLEGE OF B.A.
UNIVERSITY OF RHODE ISLAND
KINGSTON, RHODE ISLAND 02881

ROGER L. MILLS
TRW SYSTEMS GROUP
REDONDO BEACH, CAL.

STANLEY M. NAFTALY
INFORMATION MANAGEMENT INC.
432 CLAY STREET
SAN FRANCISCO, CAL. 94111

PETER NAUR
REGNECENTRALEN
RIALTO SMALLEGADE 2 B
COPENHAGEN F, DENMARK

CAPT. JOHN W. O'GRADY
127 PATTERSON RD.
BEDFORD, MASS.

ROBERT L. PATRICK
9935 DONNA
NORTHRIDGE, CAL.

BARY W. POLLACK
COMPUTATION CENTER
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305

S. L. POLLACK
NAA S+ID
MAILING CODE EA18 D/200-130
DOWNEY, CAL. 90241

BARBARA H. REMPE, AKE3
DOUGLAS AIRCRAFT
3000 OCEAN PARK BLVD.
SANTA MONICA, CAL.

JOSEPH A. RESCA
CONTROL DATA CORP.
11428 ROCKVILLE PIKE
ROCKVILLE, MARYLAND

DR. M. DEV. ROBERTS
IBM DATA SYSTEMS DIVISION
1271 AVENUE OF THE AMERICAS
NEW YORK 20, NEW YORK

ARTHUR M. ROSENBERG
SCIENTIFIC DATA SYSTEMS
1649 SEVENTEETN STREET
SANTA MONICA, CALIFORNIA

BILL ROSENTHAL
DOUGLAS AIRCRAFT
3000 OCEAN PARK BLVD.
SANTA MONICA, CAL.

MR. ARTHUR RUBINO
ADVANCED SCIENTIFIC INSTRUMENTS
8001 BLOOMINGTON FREEWAY
MINNEAPOLIS, MINNESOTA 55420

PROF. DR. H. RUTISHAUSER
EIDG. TECHNISCHE HOCHSCHULE
UNIVERSITAETSSTRASSE 10
ZURICH, SWITZERLAND

MARSHALL SAVAGE
1644 EDGECLIFFE DRIVE
HOLLYWOOD, CALIFORNIA 90026

LEE O. SCHMIDT
BECKMAN INSTRUMENTS
2400 HARBOR BLVD.
FULLERTON, CAL.

F.W. SCHNEIDER
UCLA COMPUTING FACILITY
BOELTER HALL RM. 3532B
LOS ANGELES, CAL. 90024

CHRIS SHAW
SDC
2500 COLORADO BLVD.
SANTA MONICA, CAL. 90406

S. SHELLANS
ESSO ENGINEERING
P.O. BOX 101
FLORHAM PARK, N. J. 07932

ROBERT C. SHEPARDSON
SDS
1649 SEVENTEETH STREET
SANTA MONICA, CAL 90406

GLORIA M. SILVERN
979 TEAKWOOD ROAD
LOS ANGELES, CAL. 90049

ARNOLD D. SOBOL
11070 STRATHMORE DRIVE
WESTWOOD VILLAGE, CAL. 90024

RICHARD SOUTHWORTH
LOGICON
205 AVE. I
REDONDO BEACH, CAL.

T. SUMI
LADWP
3023 DELAWARE STREET
SANTA MONICA, CAL.

WILLIS R. UNKE
UNIVAC
2276 HIGHCREST DRIVE
ROSEVILLE, MINNESOTA 55113

PROF. DR. A. VAN WIJNGAARDEN
STICHTING MATHEMATISCH CENTRUM
2E BOERHAAVESTRAAT 49
AMSTERDAM(O), NETHERLANDS

DAVID F. WEINBERG
BLDG. R-3 RM 2162
TRW SYSTEMS GROUP
REDONDO BEACH, CAL.

ALMA K. WEINSTEIN
HUGHES AIRCRAFT BLDG. 6 MS 187
FLORENCE AND TEALE STREETS
CULVER CITY, CAL.

ROBERT R. WHITE
L.A. DEPT. OF WATER + POWER
6643 BERTRAND AVE.
RESEDA CAL.

ROGER C. WILBORN
BECKMAN INSTRUMENTS
2400 HARBOR BLVD.
FULLERTON, CAL.

NIKLAUS WIRTH
COMPUTATION CENTER
STANFORD UNIVERSITY
STANFORD CAL. 94305

TERRENCE WOLD
THE RAND CORPORATION
1700 MAIN STREET
SANTA MONICA, CAL.

M. WOODGER
AUTONOMICS DIVISION
NATIONAL PHYSICAL LABORATORY
TEDDINGTON, MIDDLESEX, ENGLAND

LYNN D. YARBROUGH
HARVARD COMPUTING CENTER
CAMBRIDGE, MASSACHUSETTS 02138