NCE & TECHNOLOGY:

COBOL INSTRUMENTATION AND DEBUGGING: A CASE STUDY

NBS Special Publication 500-26

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards

NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards[1] was established by an act of Congress March 3, 1901. The Bureau's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the Nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau consists of the Institute for Basic Standards, the Institute for Materials Research, the Institute for Applied Technology, the Institute for Computer Sciences and Technology, the Office for Information Programs, and the Office of Experimental Technology Incentives Program.

THE INSTITUTE FOR BASIC STANDARDS provides the central basis within the United States of a complete and consistent system of physical measurement; coordinates that system with measurement systems of other nations; and furnishes essential services leading to accurate and uniform physical measurements throughout the Nation's scientific community, industry, and commerce. The Institute consists of the Office of Measurement Services, and the following center and divisions:

Applied Mathematics — Electricity — Mechanics — Heat — Optical Physics — Center for Radiation Research — Laboratory Astrophysics[2] — Cryogenics[2] — Electromagnetics[2] — Time and Frequency[2].

THE INSTITUTE FOR MATERIALS RESEARCH conducts materials research leading to improved methods of measurement, standards, and data on the properties of well-characterized materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government agencies; and develops, produces, and distributes standard reference materials. The Institute consists of the Office of Standard Reference Materials, the Office of Air and Water Measurement, and the following divisions:

Analytical Chemistry — Polymers — Metallurgy — Inorganic Materials — Reactor Radiation — Physical Chemistry.

THE INSTITUTE FOR APPLIED TECHNOLOGY provides technical services developing and promoting the use of available technology; cooperates with public and private organizations in developing technological standards, codes, and test methods; and provides technical advice services, and information to Government agencies and the public. The Institute consists of the following divisions and centers:

Standards Application and Analysis — Electronic Technology — Center for Consumer Product Technology: Product Systems Analysis; Product Engineering — Center for Building Technology: Structures, Materials, and Safety; Building Environment; Technical Evaluation and Application — Center for Fire Research: Fire Science; Fire Safety Engineering.

THE INSTITUTE FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides technical services designed to aid Government agencies in improving cost effectiveness in the conduct of their programs through the selection, acquisition, and effective utilization of automatic data processing equipment; and serves as the principal focus wthin the executive branch for the development of Federal standards for automatic data processing equipment, techniques, and computer languages. The Institute consist of the following divisions:

Computer Services — Systems and Software — Computer Systems Engineering — Information Technology.

THE OFFICE OF EXPERIMENTAL TECHNOLOGY INCENTIVES PROGRAM seeks to affect public policy and process to facilitate technological change in the private sector by examining and experimenting with Government policies and practices in order to identify and remove Government-related barriers and to correct inherent market imperfections that impede the innovation process.

THE OFFICE FOR INFORMATION PROGRAMS promotes optimum dissemination and accessibility of scientific information generated within NBS; promotes the development of the National Standard Reference Data System and a system of information analysis centers dealing with the broader aspects of the National Measurement System; provides appropriate services to ensure that the NBS staff has optimum accessibility to the scientific information of the world. The Office consists of the following organizational units:

Office of Standard Reference Data — Office of Information Activities — Office of Technical Publications — Library — Office of International Standards — Office of International Relations.

---

[1] Headquarters and Laboratories at Gaithersburg, Maryland, unless otherwise noted; mailing address Washington, D.C. 20234.

[2] Located at Boulder, Colorado 80302.

COMPUTER SCIENCE & TECHNOLOGY:

COBOL Instrumentation and Debugging: A Case Study

Gordon Lyon

Systems and Software Division
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, D.C. 20234

U.S. DEPARTMENT OF COMMERCE, Juanita M. Kreps, Secretary

Dr. Sidney Harman, Under Secretary

Jordan J. Baruch, Assistant Secretary for Science and Technology

U.S. NATIONAL BUREAU OF STANDARDS, Ernest Ambler, Acting Director

Issued January 1978

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publication.

National Bureau of Standards Special Publication 500-26
Nat. Bur. Stand. (U.S.), Spec. Publ. 500-26, 33 pages (Jan. 1978)
CODEN: XNBSAV

Library of Congress Catalog Card Number: 77-608347

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1978

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402

TABLE OF CONTENTS

COBOL Instrumentation and Debugging: A Case Study

Gordon Lyon

Traditionally COBOL has been run in batch execution and debugged through abnormal terminations; indeed, whole books have been devoted to deciphering COBOL core dumps. With modern equipment there is little excuse for writing COBOL source code this way. A carefully chosen example (a COBOL program to tally reserved words and cross-reference other tokens) is used as a vehicle for an examination of a contemporary COBOL coding experience. COBOL can be written quite fast interactively, and COBOL programs can be easily tuned if good timing facilities are available. Strengths and limitations of a current debugging package are highlighted in the case study.

Key words: COBOL; execution monitoring; symbolic debugging; timesharing.

1. INTRODUCTION

Studies of program measurement have been increasingly evident in recent years. Slowly an appreciation has been rebuilding for the value of measurement in program debugging, testing, and execution tuning. A paper by Knuth [5] was quite instrumental in triggering the revival of monitoring techniques [6,8], although most of the techniques themselves had been used in early days of computing [14].

Because COBOL is widely used commercially, there are many packages available to the COBOL programmer for debugging and checkout [4]. An interactive, symbolic debugging and development package has been chosen for examination from among the many packages such as cross-reference generators, "short form" macro expanders, and "neateners." There are several reasons why this choice seems a good one: (1) COBOL users often tend to ignore the enormous power of interactive programming; (2) A symbolic interactive facility frees programmers from reading tedious memory dumps (which should be obsolete for higher level languages); (3) Timing facilities and rapid recompilation encourage a programmer to check program pieces incrementally for both correctness and execution efficiency. The implications of these points are that one should use a modern computing facility to perform program development, and in particular, one with timesharing and a good file system.

In general, the programming public seems to prefer instrumentation for monitoring run-time performance when given a choice of debugging tools. The author heard a very vocal expression of this at an open discussion session at CIPS-75 [1]. The vehicle for discussion throughout will be a program, written in COBOL, that tallies COBOL reserved words and provides a crude program cross reference on non-reserved tokens. It is a fairly interesting program, and examination of the problems in "token accounting" provides a good demonstration of COBOL strengths such as binary SEARCH and external SORT. The example serves well in that it uncovers a problem inherent in the chosen debugging package, as well as being a handy utility in its own right. A complete listing appears in Appendix A.

It is sensible to approach COBOL from an interactive perspective because timesharing, or equivalently, cheap stand-alone minicomputers, are surely the current wave. Not only is program development in batch mode slow--it is often wasteful of both machine and programmers' time. Whole dumps are taken to isolate a single variable value which is buried in octal or hexadecimal [2]. Progress can be slow; reading a memory dump is often not very productive. A terminal and a symbolic breakpoint facility are far better; the programmer can catch a run at a vital point and check variables by a query such as "INTEREST ?" The system would reply, "INTEREST $4.31" which if incorrect can be changed, say, by "SET INTEREST $4.00." All of this is done symbolically and fast. The only catch is that the programmer must have written an understandable program so that he doesn't spend forty minutes between queries. Experienced programmers appear to debug at rates fifty to three hundred percent faster when on-line [12, p.6].

2. EQUIPMENT AND SOFTWARE

The work described herein was performed using a cathode ray display terminal (CRT) linked directly to a port of a time sharing, medium sized computer. No telephone dial connections were necessary, although acoustic couplers are now readily available to allow terminals to use ordinary telephones. The advantage of direct connection was line speed, in this case, 2400 baud (roughly, bits/sec) which is convenient with a fast CRT character display.

The CRT display was twenty-four eighty-character lines. An attached keyboard was comfortable and quiet, so that a programmer could share an office without annoying others.

The chosen computer system offers a large amount of on-line disk storage, so a user has no reason to carry card decks and paper tape between runs. Indeed, no system card reader is available--only a moderate speed paper tape reader. Two distinct types of magnetic tape storage are also provided. Enough information is kept on files in the system that a user can ask that a source file (of say, COBOL) be executed; if a relocatable version dated after the last source editing exists, it is loaded and execution begins. Otherwise the more recent source code is recompiled, loaded and executed.

Programs are scheduled according to their resource demands. Because timesharing emphasizes user interaction, there is a definite penalty in the scheduling for programs which become compute-bound. Monopolyzing the CPU cannot be allowed since terminal response then suffers. The environment is excellent for programming and checkout. Longer production runs can be made in a batch (or "background") mode.

Since files are crucial to a timesharing environment, the system has several editors. The one used in programming the token scanner can list selected file lines, move sequentially through the text, or search on context. Context searching has a point worth mentioning which emphasizes differences in interactive versus batch programming styles. In batch programming it is a good idea to set variables which will be program parameters, e.g. LINEWD = 50, INCRE = 50. This can lead to some confusion if parameter TWO is later set to three. Context searching allows one to use literals freely in statements, and to find them again easily if they need changing; named parameters thereby become less important. The editor can replace whole lines, pieces of lines, or read pieces of other files into the current one. File modification is easy, even for longer programs which as card decks would be difficult to handle. For insurance the user can copy his files to small reels of magnetic tape if he desires. In any event, the system also provides a file backup to within the last week.

The debugger is invoked in lieu of a compilation or run request. The programmer can (symbolically) set program variables to new values, assign or clear breakpoints where execution will pause for interrogation, display variable values, set a paragraph name trace on, and request a histogram of time in paragraphs. These few but well chosen services are sufficient to support interactive debugging and "tuning".

3. PROGRAMMING THE TEST PROBLEM

A COBOL reserved word tallier (listed in Appendix A) was chosen as the example program. Parts of the procedure division were borrowed from another program, supplemented and rearranged. This initial cut-and-paste was easily accomplished in about fifteen minutes using the online editor. The WORKING-STORAGE SECTION could have been extremely tedious to write, consisting as it does of about three hundred and fifty FILLER items. In fact, the convenience of interactive programming greatly eased the problem. First, all the reserved key words were typed into a file and sorted. A short program was written to read the key word file and generate a file of COBOL source code (called KT in the listing in Appendix A) suitable for binary SEARCHing. When the text was checked and consistent, it was edited into the program WORKING STORAGE section.

The PROCEDURE DIVISION isolates non-blank strings of characters. Termination of a line counts as a blank, so multi-line scans are not made. Once a non-literal token is found, it is looked up in the appropriate table KTx, where x is the token length. If the item is not a token in KTx, it is written along with a line number on the file NKEYS. When input is exhausted the contents of KT are printed, and the file NKEYS is sorted and also displayed. Non-reserved words (tokens) written through data record T-REC are filtered slightly prior to transmission to NKEYS. This eliminates some extraneous text fragments. The final output is a collection of reserved word counts, statistics on non-reserved words and a cross reference table.

4. PERFORMANCE INDICATIONS

After the program began running, the histogram feature was used to investigate execution behavior. A typical report appears in Figure 1. Most notable is the fact that small sections EXT-NON-BLANK-LOOP and EXT-CHR-LOOP2 consume a preponderance of running time. These sections are, of course, entered a disproportionate number of times since they scan all input characters. Table searching (L1-L15) and printing (P1-P15) cause no problem. The external SORT of non-reserved tokens (XREF) is efficient and fast. It is interesting to note that once the frequencies of the reserved words were known, tables L1-L15 were rearranged by descending frequencies for sequential SEARCHing. There was little difference between the latter and the binary SEARCH used in Appendix A. Text for the test was the analyzer itself. Results appear below in Figure 1:

HISTORY REPORT

COBDDT HISTOGRAM FOR CSNARK

PROCEDURE LABEL	ENTRIES	CPU SECS	ELAPSED TIME
START	1	0.07	0.33
NEWST	769	4.60	14.91
REG	6530	12.22	46.64
EXTRACT-TOKEN	6530	10.89	36.54
EXT-NON-BLANK-LOOP	33653	1:09.87	4:39.78
EXT-NON-BLANK-FOUND	5769	12.35	1:03.13
EXT-CHR-LOOP	5769	9.82	29.33
EXT-CHR-LOOP2	24485	59.81	3:13.84
EXT-TOKEN-END	5433	10.30	2:03.47
EXT-N-N-LITERAL	2927	7.97	58.79
EXT-N-N-LIT-END	336	0.83	1.96
EXT-EXIT	5433	9.97	1:42.78
EXT-EXT	6530	11.60	43.31
CLASSIFY-TOKEN	5433	9.38	35.55
L1	955	4.27	10.89
L2	974	5.95	18.16
L3	1038	5.10	25.81
L4	372	2.67	15.22
L5	1259	8.17	22.89
L6	493	3.34	15.72
L7	114	0.77	3.48
L8	55	0.41	0.78
L9	104	0.54	2.51
L10	13	0.04	0.11
L11	8	0.01	0.18
L12	23	0.15	0.30
L13	9	0.06	0.08
L14	3	0.01	0.01
L15	8	0.00	0.00
NEXIT	3751	9.49	37.94
CLASTOK-EXIT	5433	9.98	26.81
P1	7	0.05	0.11
P2	24	0.16	0.41

P3	21	0.15	6.70
P4	47	0.29	7.15
P5	38	0.23	0.43
P6	37	0.19	6.61
P7	36	0.16	6.68
P8	30	0.11	0.15
P9	18	0.07	0.11
P10	18	0.06	0.06
P11	9	0.03	0.06
P12	9	0.04	6.51
P13	8	0.03	0.05
P14	3	0.01	0.01
P15	3	0.02	0.05
PP	7	0.14	6.63
XREF	1	17.26	32.28
XR	1144	14.91	5:14.80

Figure 1. Timing Report.

Examination of the paragraph EXT-NON-BLANK-LOOP underscores what can be a fundamental limitation in a program tuning package, either because of limitations imposed by the operating system or failures within the package itself. The histogram indicates that the code for skipping blanks consumes twenty-two percent of execution even though it is quite brief:

```
EXT-NON-BLANK-LOOP.
                  IF IN-COL (IN-PTR) IS NOT EQUAL TO SPACE
                      GO TO EXT-NON-BLANK-FOUND.
                  ADD 1 TO IN-PTR.
                  IF IN-PTR IS NOT GREATER THAN 72
                      GO TO EXT-NON-BLANK-LOOP.
                  MOVE 0 TO TOKEN-SIZE.
                  GO TO EXT-EXT.
```

Definition of the array IN-COL is similarly transparent:

```
            ......
            DATA RECORDS ARE CARD, RIN-COL.
01          CARD PIC X(80).
01          RIN-COL.
            02 IN-COL PIC X(1) OCCURS 80 TIMES.
```

The above sections of code were rewritten in more compact forms in an effort to improve execution. RIN-COL was redrafted as

```
01          RIN-COL.
            02 IN-COL PIC X(1) OCCURS 72 TIMES
```

```
                    INDEXED BY IN-PTR.
               02 FILLER PIC (8).
```

The corresponding procedure section EXT-NON-BLANK-LOOP was changed to

```
     EXT-NON-BLANK-LOOP.
                  SEARCH IN-COL VARYING IN-PTR AT END
                    MOVE 0 TO TOKEN-SIZE, GO TO EXT-EXT;
                  WHEN IN-COL (IN-PTR) IS NOT EQUAL TO SPACE
                    NEXT SENTENCE.
```

Histograms for the new program show immediate improvement. The new 'LOOP code is entered about one fifth as often and consumes one third the cpu time of the first version of the section. Unfortunately, the new program when run without the debugging option is about one percent slower. Even after allowing for vagaries in the system clock, this indicates that section timings in the histogram package are not compensated correctly for overhead. Fast, constantly invoked code has timings that reflect many artifacts of the timing package itself. Looking at the listing in Appendix A, the section XREF--called once for a sizable computation--is probably accurately measured. On the other hand, EXT-CHR-LOOP is heavily used for short periods, so timings for EXT-CHR-LOOP can be thought somewhat doubtful.

5. OBSERVATIONS ON TIMING

The problem of designing a good timing package is not always easily solved. Hardware can supply variations [3, 15]. Some operating systems do not allow easy access to the <u>user</u> or <u>program</u> <u>state</u>, but rather lump together system overhead such as memory management operations, device completion notices, and scheduling time. Attempts to compensate for various overheads can be frustrating, since with sufficient variance there is always the possible embarassment of having too much compensation, thereby getting negative results.

6. CONCLUSIONS

Results of programming the token recognizer in COBOL lead to specific recommendations for a COBOL environment.

6.1 Files and Interactive Support

The programming system should be interactive. A quick response allows the task to move along; the programmer is not distracted by juggling many jobs to fill up his otherwise unused time between batch runs. His train of thought remains uninterrupted.

Corollary to a requirement for interaction, the system needs a good file facility. An indication of features of such files can be found in Ritchie and Thompson[11].

6.2 Run-Time Debugging Recommendations

Any interactive debugging package should allow transactions to be carried out in terms of names defined and meaningful within the COBOL source program in question. Care should be exercised to avoid forcing a programmer to resort to absolute or relocatable addresses. Using symbolic names in the source text, the following features are recommended:

(1) SET and CLEAR program breakpoints so that program execution can be selectively halted and RESTARTed.

(2) DISPLAY and RESET of variables in a simplified notation related to their initial definition. For example RESET X = 10.341.

(3) TRACEALL and TRACEBKPTS that are set on all paragraph names or selected trace and timing points in PROCEDURE DIVISION.

These first three items are more or less self explanatory--they are designed principally to replace dependence upon memory dumps.

A second set of debugging feature recommendations is to:

(4) Provide execution timings and counts of paragraph traversals.

(5) Allow selective counts and timings independent of PROCEDURE DIVISION labels by having special debugging-only labels.

The special names for timings mentioned in (5) could be distinguished comments such as "*TIME <label>".

Benchmark test sections of COBOL should be prepared to test reliabilities of any histogram package, thus:

(6) test the accuracy of the timing package.

As the example has amply demonstrated, tests are necessary. One test must be some short piece of fast but heavily called code. Comparisons of the test section with a few extra statements thrown in will quickly show if the instrumentation has a great amount of uncompensated overhead for each tally point. To illustrate this point the heavily-used paragraph EXT-NON-BLANK-LOOP was expanded to twice its normal length and retimed. The new segment of code was:

```
EXT-NON-BLANK-LOOP.

**********************dummy code enclosed below.
        IF IN-COL (IN-PTR) IS EQUAL TO SPACE
        ADD 1 TO DUMMY,
        IF IN-PTR IS GREATER THAN 72
        MOVE Ø TO TOKEN-DUMMY.
**********************dummy code enclosed above.
        IF IN-COL (IN-PTR) IS NOT EQUAL TO SPACE
                GO TO EXT-NON-BLANK-FOUND.
        ADD 1 TO IN-PTR.
        IF IN-PTR IS NOT GREATER THEN 72
                GO TO EXT-NON-BLANK-LOOP.
        MOVE Ø TO TOKEN-SIZE.
        GO TO EXT-EXT.
```

Notice that to avoid new timing points it was necessary to write the "dummy" code without labels and GO TO´s. Indicated c.p.u. time for the above (roughly doubled) paragraph was sixty-six seconds. Since the previous run took 1:09.87, it is clear that most of the indicated time for the paragraph is <u>measurement error</u>. The timing entries for REG (12.22) and EXTRACT-TOKEN (10.89) further underscore the fact that the timing is not of very high quality.

6.3 Good Compiler Diagnostics

The last recommendation is that the compiler used for debugging be tested for accuracy of error diagnostics and recovery:

(1) have a debugging compiler that gives good diagnostics.

In those cases where the debug package is integral to the compilation support there is little a user can do. In other instances compilation is a distinct step. It may be fruitful to run programs with stock (known) errors to check recovery accuracy [7,10]. One older COBOL compiler on a timesharing service was somewhat notorious for the avalanche of clutter and confusion it emitted for trivial early errors (prior to the PROCEDURE DIVISION). Also very useful is an interpretive diagnostic compiler that can catch items such as uninitialized variables. Available for years with FORTRAN, these useful debugging translators are now commercially supported for COBOL.

7. REFERENCES

[1] Canadian Information Processing Society (CIPS) National Conference, Regina, Sask., June 1975.

[2] Boar, B.H. Abend Debugging for COBOL Programmers. (John Wiley & Sons, Inc., 1976).

[3] Gentleman, W.M. and Wickman, B.A. Timing on computers. SIGARCH Computer Architecture News 2, (October, 1973), 20-23.

[4] ICP Reference Series Software Directory Volume 1, Data Processing Management, International Computer Programs, Inc., 1976.

[5] Knuth, D.E. An empirical study of FORTRAN programs. SOFTWARE: Practice & Experience 1, (1971), 105-133.

[6] Knuth, D.E., and Stevenson, F.R. Optimal measurement points for program frequency counts. B.I.T. 13, (1973), 313-322.

[7] Litecky, C.R., and Davis, G.B. A study of errors, error-proneness, and error diagnosis in COBOL. Comm. ACM. 19, 1(January, 1976), 33-37.

[8] Lyon, G.E., and Stillman, R.B. Simple transforms for instrumenting FORTRAN decks. SOFTWARE: Practice & Experience 5, (1975), 347-358.

[9] Operator's Handbook--OMRON 8025. OMRON Information Products Division, Sunnyvale, Calif., 1974.

[10] Presser, L. and Benson, J. Evaluation of compiler statistics. Computer Journal 17, 2(1974), 121-123.

[11] Ritchie, D.M., and Thompson, K. The UNIX time-sharing system. Comm. ACM 17, 7(July, 1974), 365-375.

[12] Sackman, H., et.al. Exploratory experimental studies comparing online and offline programming performance. Comm. ACM. 11, 1(January 1968), 3-11.

[13] Thompson, K. and Ritchie, D.M. UNIX Programmer's Manual. Bell Telephone Laboratories, 1975. See the command "cbd" for a debugging package that requires little advance planning.

[14] Wilkes, M. Software engineering and structured programming. IEEE Transactions on Software Engineering SE-2, 4(December, 1976), 274-276.

[15] Wortman, David B. A study of high-resolution timing. IEEE Transactions on Software Engineering SE-2, 2(June, 1976), 135-137.

8. APPENDIX A

```
     IDENTIFICATION DIVISION.
*        KEYWORD SEEKER AND XREF SEPT. 10, 1975.
*********** THIS VERSION FILTERS N-R´S , SEE "NEXIT".*****
PROGRAM-ID. CSNARK.
REMARKS.  TALLYS RESERVED WORDS. SOME STAT ON NON-RES´D.

ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
        SELECT NKEYS ASSIGN TO DSK RECORDING MODE IS ASCII.
        SELECT TXT ASSIGN TO DSK RECORDING MODE IS ASCII.
        SELECT SORT-FILE ASSIGN TO DSK, DSK, DSK, DSK, DSK.

DATA DIVISION.
FILE SECTION.
SD  SORT-FILE.
01  SORT-RECORD.
        02 A PIC X(15).
        02 B PIC S9(5).
FD TXT
        LABEL RECORDS ARE STANDARD
        VALUE OF ID IS "ABC      "
        DATA RECORDS ARE CARD, RIN-COL.
01      CARD PIC X(80).
01      RIN-COL.
        02 IN-COL PIC X OCCURS 80 TIMES.
FD  NKEYS
        LABEL RECORDS ARE STANDARD
        VALUE OF ID IS "NKEYS TST"
        DATA RECORD IS T-REC.
01   T-REC.
        02  TTOKK PIC X(15).
        02  TLEN PIC S9(4)   .
WORKING-STORAGE SECTION.
01      KW-STAT.
        02 KW1 PIC S9(5)  OCCURS 7 TIMES.
        02 KW2 PIC S9(5)  OCCURS 24 TIMES.
        02 KW3 PIC S9(5)  OCCURS 21 TIMES.
        02 KW4 PIC S9(5)  OCCURS 47 TIMES.
        02 KW5 PIC S9(5)  OCCURS 38 TIMES.
        02 KW6 PIC S9(5)  OCCURS 37 TIMES.
        02 KW7 PIC S9(5)  OCCURS 36 TIMES.
        02 KW8 PIC S9(5)  OCCURS 30 TIMES.
        02 KW9 PIC S9(5)  OCCURS 18 TIMES.
        02 KW10 PIC S9(5)  OCCURS 18 TIMES.
        02 KW11 PIC S9(5)  OCCURS 9 TIMES.
        02 KW12 PIC S9(5)  OCCURS 9 TIMES.
        02 KW13 PIC S9(5)  OCCURS 8 TIMES.
        02 KW14 PIC S9(5)  OCCURS 3 TIMES.
```

```
       02 KW15 PIC S9(5)  OCCURS 3 TIMES.
*      TABLES BELOW ARE MACRO-EXPANSIONS
*      FROM TABLE-GENERATING CODE WRITTEN
*      INTERACTIVELY IN Algol-60.
01     KT.
       02 KT1.
             03 FILLER PIC X(1)  VALUE "*".
             03 FILLER PIC X(1)  VALUE "+".
             03 FILLER PIC X(1)  VALUE "-".
             03 FILLER PIC X(1)  VALUE "/".
             03 FILLER PIC X(1)  VALUE "<".
             03 FILLER PIC X(1)  VALUE "=".
             03 FILLER PIC X(1)  VALUE ">".
       02 STR1 REDEFINES KT1 OCCURS 7 TIMES
             INDEXED BY TABINX, ASCENDING KEY K1.
             04 K1 PIC X(1).
       02 KT2.
             03 FILLER PIC X(2)  VALUE "**".
             03 FILLER PIC X(2)  VALUE "AT".
             03 FILLER PIC X(2)  VALUE "BY".
             03 FILLER PIC X(2)  VALUE "CD".
             03 FILLER PIC X(2)  VALUE "CF".
             03 FILLER PIC X(2)  VALUE "CH".
             03 FILLER PIC X(2)  VALUE "DE".
             03 FILLER PIC X(2)  VALUE "FD".
             03 FILLER PIC X(2)  VALUE "GO".
             03 FILLER PIC X(2)  VALUE "IF".
             03 FILLER PIC X(2)  VALUE "IN".
             03 FILLER PIC X(2)  VALUE "IS".
             03 FILLER PIC X(2)  VALUE "NO".
             03 FILLER PIC X(2)  VALUE "OF".
             03 FILLER PIC X(2)  VALUE "ON".
             03 FILLER PIC X(2)  VALUE "OR".
             03 FILLER PIC X(2)  VALUE "PF".
             03 FILLER PIC X(2)  VALUE "PH".
             03 FILLER PIC X(2)  VALUE "RD".
             03 FILLER PIC X(2)  VALUE "RF".
             03 FILLER PIC X(2)  VALUE "RH".
             03 FILLER PIC X(2)  VALUE "SD".
             03 FILLER PIC X(2)  VALUE "TO".
             03 FILLER PIC X(2)  VALUE "UP".
       02 STR2 REDEFINES KT2 OCCURS 24 TIMES
             INDEXED BY TABINX, ASCENDING KEY K2.
             04 K2 PIC X(2).
       02 KT3.
             03 FILLER PIC X(3)  VALUE "ADD".
             03 FILLER PIC X(3)  VALUE "ALL".
             03 FILLER PIC X(3)  VALUE "AND".
             03 FILLER PIC X(3)  VALUE "ARE".
             03 FILLER PIC X(3)  VALUE "DAY".
```

```
           03 FILLER PIC X(3)  VALUE "EGI".
           03 FILLER PIC X(3)  VALUE "EMI".
           03 FILLER PIC X(3)  VALUE "END".
           03 FILLER PIC X(3)  VALUE "EOP".
           03 FILLER PIC X(3)  VALUE "ESI".
           03 FILLER PIC X(3)  VALUE "FOR".
           03 FILLER PIC X(3)  VALUE "I-O".
           03 FILLER PIC X(3)  VALUE "KEY".
           03 FILLER PIC X(3)  VALUE "NOT".
           03 FILLER PIC X(3)  VALUE "OFF".
           03 FILLER PIC X(3)  VALUE "PIC".
           03 FILLER PIC X(3)  VALUE "RUN".
           03 FILLER PIC X(3)  VALUE "SET".
           03 FILLER PIC X(3)  VALUE "SUM".
           03 FILLER PIC X(3)  VALUE "TOP".
           03 FILLER PIC X(3)  VALUE "USE".
       02 STR3 REDEFINES KT3 OCCURS 21 TIMES
           INDEXED BY TABINX, ASCENDING KEY K3.
           04 K3 PIC X(3).
       02 KT4.
           03 FILLER PIC X(4)  VALUE "ALSO".
           03 FILLER PIC X(4)  VALUE "AREA".
           03 FILLER PIC X(4)  VALUE "CALL".
           03 FILLER PIC X(4)  VALUE "CODE".
           03 FILLER PIC X(4)  VALUE "COMP".
           03 FILLER PIC X(4)  VALUE "COPY".
           03 FILLER PIC X(4)  VALUE "CORR".
           03 FILLER PIC X(4)  VALUE "DATA".
           03 FILLER PIC X(4)  VALUE "DATE".
           03 FILLER PIC X(4)  VALUE "DOWN".
           03 FILLER PIC X(4)  VALUE "ELSE".
           03 FILLER PIC X(4)  VALUE "EXIT".
           03 FILLER PIC X(4)  VALUE "FILE".
           03 FILLER PIC X(4)  VALUE "FROM".
           03 FILLER PIC X(4)  VALUE "INTO".
           03 FILLER PIC X(4)  VALUE "JUST".
           03 FILLER PIC X(4)  VALUE "LAST".
           03 FILLER PIC X(4)  VALUE "LEFT".
           03 FILLER PIC X(4)  VALUE "LESS".
           03 FILLER PIC X(4)  VALUE "LINE".
           03 FILLER PIC X(4)  VALUE "LOCK".
           03 FILLER PIC X(4)  VALUE "MODE".
           03 FILLER PIC X(4)  VALUE "MOVE".
           03 FILLER PIC X(4)  VALUE "NEXT".
           03 FILLER PIC X(4)  VALUE "OPEN".
           03 FILLER PIC X(4)  VALUE "PAGE".
           03 FILLER PIC X(4)  VALUE "PLUS".
           03 FILLER PIC X(4)  VALUE "READ".
           03 FILLER PIC X(4)  VALUE "REEL".
           03 FILLER PIC X(4)  VALUE "SAME".
           03 FILLER PIC X(4)  VALUE "SEND".
           03 FILLER PIC X(4)  VALUE "SIGN".
```

```
           03 FILLER PIC X(4)  VALUE "SIZE".
           03 FILLER PIC X(4)  VALUE "SORT".
           03 FILLER PIC X(4)  VALUE "STOP".
           03 FILLER PIC X(4)  VALUE "SYNC".
           03 FILLER PIC X(4)  VALUE "TAPE".
           03 FILLER PIC X(4)  VALUE "TEXT".
           03 FILLER PIC X(4)  VALUE "THAN".
           03 FILLER PIC X(4)  VALUE "THRU".
           03 FILLER PIC X(4)  VALUE "TIME".
           03 FILLER PIC X(4)  VALUE "TYPE".
           03 FILLER PIC X(4)  VALUE "UNIT".
           03 FILLER PIC X(4)  VALUE "UPON".
           03 FILLER PIC X(4)  VALUE "WHEN".
           03 FILLER PIC X(4)  VALUE "WITH".
           03 FILLER PIC X(4)  VALUE "ZERO".
02 STR4 REDEFINES KT4 OCCURS 47 TIMES
           INDEXED BY TABINX, ASCENDING KEY K4.
           04 K4 PIC X(4).
02 KT5.
           03 FILLER PIC X(5)  VALUE "AFTER".
           03 FILLER PIC X(5)  VALUE "ALTER".
           03 FILLER PIC X(5)  VALUE "AREAS".
           03 FILLER PIC X(5)  VALUE "BLANK".
           03 FILLER PIC X(5)  VALUE "BLOCK".
           03 FILLER PIC X(5)  VALUE "CLOSE".
           03 FILLER PIC X(5)  VALUE "COBOL".
           03 FILLER PIC X(5)  VALUE "COMMA".
           03 FILLER PIC X(5)  VALUE "COUNT".
           03 FILLER PIC X(5)  VALUE "ENTER".
           03 FILLER PIC X(5)  VALUE "EQUAL".
           03 FILLER PIC X(5)  VALUE "ERROR".
           03 FILLER PIC X(5)  VALUE "EVERY".
           03 FILLER PIC X(5)  VALUE "FINAL".
           03 FILLER PIC X(5)  VALUE "FIRST".
           03 FILLER PIC X(5)  VALUE "GROUP".
           03 FILLER PIC X(5)  VALUE "INDEX".
           03 FILLER PIC X(5)  VALUE "INPUT".
           03 FILLER PIC X(5)  VALUE "LABEL".
           03 FILLER PIC X(5)  VALUE "LIMIT".
           03 FILLER PIC X(5)  VALUE "LINES".
           03 FILLER PIC X(5)  VALUE "MERGE".
           03 FILLER PIC X(5)  VALUE "QUEUE".
           03 FILLER PIC X(5)  VALUE "QUOTE".
           03 FILLER PIC X(5)  VALUE "RERUN".
           03 FILLER PIC X(5)  VALUE "RESET".
           03 FILLER PIC X(5)  VALUE "RIGHT".
           03 FILLER PIC X(5)  VALUE "SPACE".
           03 FILLER PIC X(5)  VALUE "START".
           03 FILLER PIC X(5)  VALUE "TABLE".
           03 FILLER PIC X(5)  VALUE "TIMES".
           03 FILLER PIC X(5)  VALUE "UNTIL".
           03 FILLER PIC X(5)  VALUE "USAGE".
```

```
        03 FILLER PIC X(5)  VALUE "USING".
        03 FILLER PIC X(5)  VALUE "VALUE".
        03 FILLER PIC X(5)  VALUE "WORDS".
        03 FILLER PIC X(5)  VALUE "WRITE".
        03 FILLER PIC X(5)  VALUE "ZEROS".
02 STR5 REDEFINES KT5 OCCURS 38 TIMES
        INDEXED BY TABINX, ASCENDING KEY K5.
        04 K5 PIC X(5).
02 KT6.
        03 FILLER PIC X(6)  VALUE "ACCEPT".
        03 FILLER PIC X(6)  VALUE "ACCESS".
        03 FILLER PIC X(6)  VALUE "ASSIGN".
        03 FILLER PIC X(6)  VALUE "AUTHOR".
        03 FILLER PIC X(6)  VALUE "BEFORE".
        03 FILLER PIC X(6)  VALUE "BOTTOM".
        03 FILLER PIC X(6)  VALUE "CANCEL".
        03 FILLER PIC X(6)  VALUE "COLUMN".
        03 FILLER PIC X(6)  VALUE "DELETE".
        03 FILLER PIC X(6)  VALUE "DETAIL".
        03 FILLER PIC X(6)  VALUE "DIVIDE".
        03 FILLER PIC X(6)  VALUE "ENABLE".
        03 FILLER PIC X(6)  VALUE "EXTEND".
        03 FILLER PIC X(6)  VALUE "FILLER".
        03 FILLER PIC X(6)  VALUE "GIVING".
        03 FILLER PIC X(6)  VALUE "LENGTH".
        03 FILLER PIC X(6)  VALUE "LIMITS".
        03 FILLER PIC X(6)  VALUE "LINAGE".
        03 FILLER PIC X(6)  VALUE "MEMORY".
        03 FILLER PIC X(6)  VALUE "NATIVE".
        03 FILLER PIC X(6)  VALUE "NUMBER".
        03 FILLER PIC X(6)  VALUE "OCCURS".
        03 FILLER PIC X(6)  VALUE "OUTPUT".
        03 FILLER PIC X(6)  VALUE "QUOTES".
        03 FILLER PIC X(6)  VALUE "RANDOM".
        03 FILLER PIC X(6)  VALUE "RECORD".
        03 FILLER PIC X(6)  VALUE "REPORT".
        03 FILLER PIC X(6)  VALUE "RETURN".
        03 FILLER PIC X(6)  VALUE "REWIND".
        03 FILLER PIC X(6)  VALUE "SEARCH".
        03 FILLER PIC X(6)  VALUE "SELECT".
        03 FILLER PIC X(6)  VALUE "SOURCE".
        03 FILLER PIC X(6)  VALUE "SPACES".
        03 FILLER PIC X(6)  VALUE "STATUS".
        03 FILLER PIC X(6)  VALUE "STRING".
        03 FILLER PIC X(6)  VALUE "VALUES".
        03 FILLER PIC X(6)  VALUE "ZEROES".
02 STR6 REDEFINES KT6 OCCURS 37 TIMES
        INDEXED BY TABINX, ASCENDING KEY K6.
        04 K6 PIC X(6).
02 KT7.
        03 FILLER PIC X(7)  VALUE "COMPUTE".
        03 FILLER PIC X(7)  VALUE "CONTROL".
```

```
          03 FILLER PIC X(7)  VALUE "DISABLE".
          03 FILLER PIC X(7)  VALUE "DISPLAY".
          03 FILLER PIC X(7)  VALUE "DYNAMIC".
          03 FILLER PIC X(7)  VALUE "FOOTING".
          03 FILLER PIC X(7)  VALUE "GREATER".
          03 FILLER PIC X(7)  VALUE "HEADING".
          03 FILLER PIC X(7)  VALUE "INDEXED".
          03 FILLER PIC X(7)  VALUE "INITIAL".
          03 FILLER PIC X(7)  VALUE "INSPECT".
          03 FILLER PIC X(7)  VALUE "INVALID".
          03 FILLER PIC X(7)  VALUE "LEADING".
          03 FILLER PIC X(7)  VALUE "LINKAGE".
          03 FILLER PIC X(7)  VALUE "MESSAGE".
          03 FILLER PIC X(7)  VALUE "MODULES".
          03 FILLER PIC X(7)  VALUE "NUMERIC".
          03 FILLER PIC X(7)  VALUE "OMITTED".
          03 FILLER PIC X(7)  VALUE "PERFORM".
          03 FILLER PIC X(7)  VALUE "PICTURE".
          03 FILLER PIC X(7)  VALUE "POINTER".
          03 FILLER PIC X(7)  VALUE "PROCEED".
          03 FILLER PIC X(7)  VALUE "PROGRAM".
          03 FILLER PIC X(7)  VALUE "RECEIVE".
          03 FILLER PIC X(7)  VALUE "RECORDS".
          03 FILLER PIC X(7)  VALUE "RELEASE".
          03 FILLER PIC X(7)  VALUE "REMOVAL".
          03 FILLER PIC X(7)  VALUE "RENAMES".
          03 FILLER PIC X(7)  VALUE "REPORTS".
          03 FILLER PIC X(7)  VALUE "RESERVE".
          03 FILLER PIC X(7)  VALUE "REWRITE".
          03 FILLER PIC X(7)  VALUE "ROUNDED".
          03 FILLER PIC X(7)  VALUE "SECTION".
          03 FILLER PIC X(7)  VALUE "SEGMENT".
          03 FILLER PIC X(7)  VALUE "THROUGH".
          03 FILLER PIC X(7)  VALUE "VARYING".
     02 STR7 REDEFINES KT7 OCCURS 36 TIMES
          INDEXED BY TABINX, ASCENDING KEY K7.
          04 K7 PIC X(7).
     02 KT8.
          03 FILLER PIC X(8)  VALUE "CODE-SET".
          03 FILLER PIC X(8)  VALUE "CONTAINS".
          03 FILLER PIC X(8)  VALUE "CONTROLS".
          03 FILLER PIC X(8)  VALUE "CURRENCY".
          03 FILLER PIC X(8)  VALUE "DIVISION".
          03 FILLER PIC X(8)  VALUE "GENERATE".
          03 FILLER PIC X(8)  VALUE "INDICATE".
          03 FILLER PIC X(8)  VALUE "INITIATE".
          03 FILLER PIC X(8)  VALUE "MULTIPLE".
          03 FILLER PIC X(8)  VALUE "MULTIPLY".
          03 FILLER PIC X(8)  VALUE "NEGATIVE".
          03 FILLER PIC X(8)  VALUE "OPTIONAL".
          03 FILLER PIC X(8)  VALUE "OVERFLOW".
          03 FILLER PIC X(8)  VALUE "POSITION".
```

```
          03 FILLER PIC X(8)  VALUE "POSITIVE".
          03 FILLER PIC X(8)  VALUE "PRINTING".
          03 FILLER PIC X(8)  VALUE "RELATIVE".
          03 FILLER PIC X(8)  VALUE "REVERSED".
          03 FILLER PIC X(8)  VALUE "SECURITY".
          03 FILLER PIC X(8)  VALUE "SENTENCE".
          03 FILLER PIC X(8)  VALUE "SEPARATE".
          03 FILLER PIC X(8)  VALUE "SEQUENCE".
          03 FILLER PIC X(8)  VALUE "STANDARD".
          03 FILLER PIC X(8)  VALUE "SUBTRACT".
          03 FILLER PIC X(8)  VALUE "SUPPRESS".
          03 FILLER PIC X(8)  VALUE "SYMBOLIC".
          03 FILLER PIC X(8)  VALUE "TALLYING".
          03 FILLER PIC X(8)  VALUE "TERMINAL".
          03 FILLER PIC X(8)  VALUE "TRAILING".
          03 FILLER PIC X(8)  VALUE "UNSTRING".
     02 STR8 REDEFINES KT8 OCCURS 30 TIMES
          INDEXED BY TABINX, ASCENDING KEY K8.
          04 K8 PIC X(8).
     02 KT9.
          03 FILLER PIC X(9)  VALUE "ADVANCING".
          03 FILLER PIC X(9)  VALUE "ALTERNATE".
          03 FILLER PIC X(9)  VALUE "ASCENDING".
          03 FILLER PIC X(9)  VALUE "CHARACTER".
          03 FILLER PIC X(9)  VALUE "COLLATING".
          03 FILLER PIC X(9)  VALUE "DEBUGGING".
          03 FILLER PIC X(9)  VALUE "DELIMITED".
          03 FILLER PIC X(9)  VALUE "DELIMITER".
          03 FILLER PIC X(9)  VALUE "DEPENDING".
          03 FILLER PIC X(9)  VALUE "EXCEPTION".
          03 FILLER PIC X(9)  VALUE "JUSTIFIED".
          03 FILLER PIC X(9)  VALUE "LOW-VALUE".
          03 FILLER PIC X(9)  VALUE "PROCEDURE".
          03 FILLER PIC X(9)  VALUE "REDEFINES".
          03 FILLER PIC X(9)  VALUE "REMAINDER".
          03 FILLER PIC X(9)  VALUE "REPLACING".
          03 FILLER PIC X(9)  VALUE "REPORTING".
          03 FILLER PIC X(9)  VALUE "TERMINATE".
     02 STR9 REDEFINES KT9 OCCURS 18 TIMES
          INDEXED BY TABINX, ASCENDING KEY K9.
          04 K9 PIC X(9).
     02 KT10.
          03 FILLER PIC X(10)  VALUE "ALPHABETIC".
          03 FILLER PIC X(10)  VALUE "CHARACTERS".
          03 FILLER PIC X(10)  VALUE "DEBUG-ITEM".
          03 FILLER PIC X(10)  VALUE "DEBUG-LINE".
          03 FILLER PIC X(10)  VALUE "DEBUG-NAME".
          03 FILLER PIC X(10)  VALUE "DEBUG-SUB1".
          03 FILLER PIC X(10)  VALUE "DEBUG-SUB2".
          03 FILLER PIC X(10)  VALUE "DEBUG-SUB3".
          03 FILLER PIC X(10)  VALUE "DESCENDING".
          03 FILLER PIC X(10)  VALUE "DUPLICATES".
```

```
         03 FILLER PIC X(10)  VALUE "HIGH-VALUE".
         03 FILLER PIC X(10)  VALUE "LOW-VALUES".
         03 FILLER PIC X(10)  VALUE "PROCEDURES".
         03 FILLER PIC X(10)  VALUE "PROGRAM-ID".
         03 FILLER PIC X(10)  VALUE "REFERENCES".
         03 FILLER PIC X(10)  VALUE "SEQUENTIAL".
         03 FILLER PIC X(10)  VALUE "SORT-MERGE".
         03 FILLER PIC X(10)  VALUE "STANDARD-1".
   02 STR10 REDEFINES KT10 OCCURS 18 TIMES
         INDEXED BY TABINX, ASCENDING KEY K10.
         04 K10 PIC X(10).
   02 KT11.
         03 FILLER PIC X(11)  VALUE "CLOCK-UNITS".
         03 FILLER PIC X(11)  VALUE "DESTINATION".
         03 FILLER PIC X(11)  VALUE "END-OF-PAGE".
         03 FILLER PIC X(11)  VALUE "ENVIRONMENT".
         03 FILLER PIC X(11)  VALUE "HIGH-VALUES".
         03 FILLER PIC X(11)  VALUE "I-O-CONTROL".
         03 FILLER PIC X(11)  VALUE "SUB-QUEUE-1".
         03 FILLER PIC X(11)  VALUE "SUB-QUEUE-2".
         03 FILLER PIC X(11)  VALUE "SUB-QUEUE-3".
   02 STR11 REDEFINES KT11 OCCURS 9 TIMES
         INDEXED BY TABINX, ASCENDING KEY K11.
         04 K11 PIC X(11).
   02 KT12.
         03 FILLER PIC X(12)  VALUE "DATE-WRITTEN".
         03 FILLER PIC X(12)  VALUE "DECLARATIVES".
         03 FILLER PIC X(12)  VALUE "FILE-CONTROL".
         03 FILLER PIC X(12)  VALUE "INPUT-OUTPUT".
         03 FILLER PIC X(12)  VALUE "INSTALLATION".
         03 FILLER PIC X(12)  VALUE "LINE-COUNTER".
         03 FILLER PIC X(12)  VALUE "ORGANIZATION".
         03 FILLER PIC X(12)  VALUE "PAGE-COUNTER".
         03 FILLER PIC X(12)  VALUE "SYNCHRONIZED".
   02 STR12 REDEFINES KT12 OCCURS 9 TIMES
         INDEXED BY TABINX, ASCENDING KEY K12.
         04 K12 PIC X(12).
   02 KT13.
         03 FILLER PIC X(13)  VALUE "COMMUNICATION".
         03 FILLER PIC X(13)  VALUE "COMPUTATIONAL".
         03 FILLER PIC X(13)  VALUE "CONFIGURATION".
         03 FILLER PIC X(13)  VALUE "CORRESPONDING".
         03 FILLER PIC X(13)  VALUE "DATE-COMPILED".
         03 FILLER PIC X(13)  VALUE "DECIMAL-POINT".
         03 FILLER PIC X(13)  VALUE "SEGMENT-LIMIT".
         03 FILLER PIC X(13)  VALUE "SPECIAL-NAMES".
   02 STR13 REDEFINES KT13 OCCURS 8 TIMES
         INDEXED BY TABINX, ASCENDING KEY K13.
         04 K13 PIC X(13).
   02 KT14.
         03 FILLER PIC X(14)  VALUE "DEBUG-CONTENTS".
         03 FILLER PIC X(14)  VALUE "IDENTIFICATION".
```

```
                03 FILLER PIC X(14)  VALUE "LINAGE-COUNTER".
        02 STR14 REDEFINES KT14 OCCURS 3 TIMES
                INDEXED BY TABINX, ASCENDING KEY K14.
                04 K14 PIC X(14).
        02 KT15.
                03 FILLER PIC X(15)  VALUE "OBJECT-COMPUTER".
                03 FILLER PIC X(15)  VALUE "SOURCE-COMPUTER".
                03 FILLER PIC X(15)  VALUE "WORKING-STORAGE".
        02 STR15 REDEFINES KT15 OCCURS 3 TIMES
                INDEXED BY TABINX, ASCENDING KEY K15.
                04 K15 PIC X(15).
01 TOKEN-RECORD.
        02 TOKEN-LEN PIC S9(5) COMP SYNC.
        02 TOKEN-SIZE PIC S9(5) COMP SYNC.
        02 TOKEN.
                03 TOKEN-CHR PIC X OCCURS 50 TIMES.
                03 TOKEN-PUNC PIC X.
01 TMASK-RECORD REDEFINES TOKEN-RECORD.
        02 TMASK-LEN PIC S9(5) COMP SYNC.
        02 TMASK-SIZE PIC S9(5) COMP SYNC.
        02 TMASK.
                03 TMASK-CHR PIC X(50).
                03  TT1 REDEFINES TMASK-CHR.
                        04 T1 PIC X(1).
                        04 FILLER PIC X(49).
                03  TT2 REDEFINES TMASK-CHR.
                        04 T2 PIC X(2).
                        04 FILLER PIC X(48).
                03  TT3 REDEFINES TMASK-CHR.
                        04 T3 PIC X(3).
                        04 FILLER PIC X(47).
                03  TT4 REDEFINES TMASK-CHR.
                        04 T4 PIC X(4).
                        04 FILLER PIC X(46).
                03  TT5 REDEFINES TMASK-CHR.
                        04 T5 PIC X(5).
                        04 FILLER PIC X(45).
                03  TT6 REDEFINES TMASK-CHR.
                        04 T6 PIC X(6).
                        04 FILLER PIC X(44).
                03  TT7 REDEFINES TMASK-CHR.
                        04 T7 PIC X(7).
                        04 FILLER PIC X(43).
                03  TT8 REDEFINES TMASK-CHR.
                        04 T8 PIC X(8).
                        04 FILLER PIC X(42).
                03  TT9 REDEFINES TMASK-CHR.
                        04 T9 PIC X(9).
                        04 FILLER PIC X(41).
                03  TT10 REDEFINES TMASK-CHR.
                        04 T10 PIC X(10).
                        04 FILLER PIC X(40).
```

```
                03  TT11 REDEFINES TMASK-CHR.
                        04 T11 PIC X(11).
                        04 FILLER PIC X(39).
                03  TT12 REDEFINES TMASK-CHR.
                        04 T12 PIC X(12).
                        04 FILLER PIC X(38).
                03  TT13 REDEFINES TMASK-CHR.
                        04 T13 PIC X(13).
                        04 FILLER PIC X(37).
                03  TT14 REDEFINES TMASK-CHR.
                        04 T14 PIC X(14).
                        04 FILLER PIC X(36).
                03  TT15 REDEFINES TMASK-CHR.
                        04 T15 PIC X(15).
                        04 FILLER PIC X(35).
                03 TMASK-PUNC PIC X.
01  MISC-STAT.
        02 NSENT PIC S9(10) COMP SYNC.
        02 NRESW PIC S9(10) COMP SYNC.
        02 NNRESW PIC S9(10) COMP SYNC.
        02 SUMNRL PIC S9(10) COMP SYNC.
        02 SUMNRL2 PIC S9(10) COMP SYNC.
        02 CARDCT PIC S9(4) COMP SYNC.
01 OUT-TEXT.
        02 O-ST.
           03 FILLER PIC X(40)
              VALUE  " SENTENCES: PERIOD TERMINATES .".
           03 FILLER PIC X(40)
              VALUE  " RESERVED WORDS IN ALL.".
           03 FILLER PIC X(40)
              VALUE  " NON-RESERVED TOKENS, CALLED N-R BELOW.".
           03 FILLER PIC X(40)
              VALUE  " AVERAGE N-R LENGTH.".
           03 FILLER PIC X(40)
              VALUE  " VARIANCE IN LENGTH OF N-R TOKENS.".
           03 FILLER PIC X(40)
              VALUE  " TOTAL RESVD WORDS AND N-R TOKENS.".
           03 FILLER PIC X(40)
              VALUE  " CARD IMAGES READ.".
        02 O-SM REDEFINES O-ST PIC X(40)
                OCCURS 7 TIMES INDEXED BY I.
        02 O-S PIC S9(10)V9999
                OCCURS 7 TIMES INDEXED BY I.
01  PLINE.
        02 L PIC X(51).
        02 LXL REDEFINES L.
                03 P-TOK PIC X(15).
                03 L-REST OCCURS 9 TIMES INDEXED BY PCTR.
                        04 L-FIELD PIC Z(4).

77  PCTR PIC S9(7) COMP SYNC.
77  P-TOK2 PIC X(15).
```

```
77  IN-CHR PIC X.
77  IN-PTR PIC S9(5) COMP SYNC.
77  I PIC S9(5) COMP SYNC.
77  TOKEN-BEG PIC S9(5) COMP SYNC.
77  TOKEN-PTR PIC S9(5) COMP SYNC.
77  TABINX, PIC S99, COMPUTATIONAL SYNC.
77  TEMPR PIC S9(10) COMP SYNC.
77  QT0 PIC S9(5)V99 COMP SYNC.
77  QT1 PIC S9(10)V9999 COMP SYNC.
77  QT2 PIC S9(10)V9999 COMP SYNC.

PROCEDURE DIVISION.


*       Note: Use of DISPLAY is system-dependent,
*       as is COMP SYNC above.


START.
*       MISC-STAT (COMP SYNC) AUTO SET TO 0´S
        MOVE ZEROES TO KW-STAT.
        OPEN OUTPUT NKEYS.
        OPEN INPUT TXT.
NEWST.
        MOVE 1 TO IN-PTR.
        READ TXT RECORD INTO CARD AT END
                PERFORM P1 VARYING I FROM 1 BY 1 UNTIL I > 7,
                PERFORM P2 VARYING I FROM 1 BY 1 UNTIL I > 24,
                PERFORM P3 VARYING I FROM 1 BY 1 UNTIL I > 21,
                PERFORM P4 VARYING I FROM 1 BY 1 UNTIL I > 47,
                PERFORM P5 VARYING I FROM 1 BY 1 UNTIL I > 38,
                PERFORM P6 VARYING I FROM 1 BY 1 UNTIL I > 37,
                PERFORM P7 VARYING I FROM 1 BY 1 UNTIL I > 36,
                PERFORM P8 VARYING I FROM 1 BY 1 UNTIL I > 30,
                PERFORM P9 VARYING I FROM 1 BY 1 UNTIL I > 18,
                PERFORM P10 VARYING I FROM 1 BY 1 UNTIL I > 18,
                PERFORM P11 VARYING I FROM 1 BY 1 UNTIL I > 9,
                PERFORM P12 VARYING I FROM 1 BY 1 UNTIL I > 9,
                PERFORM P13 VARYING I FROM 1 BY 1 UNTIL I > 8,
                PERFORM P14 VARYING I FROM 1 BY 1 UNTIL I > 3,
                PERFORM P15 VARYING I FROM 1 BY 1 UNTIL I > 3,
                MOVE NSENT TO O-S (1),
                MOVE NRESW TO O-S (2),
                MOVE NNRESW TO O-S (3),
                DIVIDE SUMNRL BY NNRESW GIVING QT0,
                MOVE QT0 TO O-S (4),
                MULTIPLY QT0 BY QT0 GIVING QT1,
                DIVIDE SUMNRL2 BY NNRESW GIVING QT2,
                SUBTRACT QT1 FROM QT2 GIVING O-S (5),
                ADD NRESW, NNRESW GIVING O-S (6),
                MOVE CARDCT TO O-S (7),
                PERFORM PP VARYING I FROM 1 BY 1 UNTIL I > 7,
```

```
                PERFORM XREF,
                GO TO N-TERM.
           ADD 1 TO CARDCT.
      *                  DISPLAY CARD.
       REG.
           PERFORM EXTRACT-TOKEN.
           IF IN-PTR IS GREATER THAN 72 GO TO NEWST
           ELSE GO TO REG.
       EXTRACT-TOKEN SECTION.
       EXT-NON-BLANK-LOOP.
           IF IN-COL (IN-PTR) IS NOT EQUAL TO SPACE
                GO TO EXT-NON-BLANK-FOUND.
           ADD 1 TO IN-PTR.
           IF IN-PTR IS NOT GREATER THAN 72
                GO TO EXT-NON-BLANK-LOOP.
           MOVE Ø TO TOKEN-SIZE.
           GO TO EXT-EXT.
       EXT-NON-BLANK-FOUND.
           MOVE IN-COL (IN-PTR) TO IN-CHR.
           MOVE IN-PTR TO TOKEN-BEG.
           MOVE SPACES TO TOKEN.
           MOVE 1 TO I.
           MOVE Ø TO TOKEN-PTR.
       EXT-CHR-LOOP.
           IF IN-CHR IS EQUAL TO QUOTE
                  GO TO EXT-N-N-LITERAL.
       EXT-CHR-LOOP2.
           IF IN-CHR IS EQUAL TO SPACE
           OR IN-CHR IS EQUAL TO ´,´ OR
           IN-CHR IS EQUAL TO ´;´
           OR IN-CHR IS EQUAL TO ´.´
                ADD 1 TO IN-PTR
                GO TO EXT-TOKEN-END.
           ADD 1 TO TOKEN-PTR.
           MOVE IN-CHR TO TOKEN-CHR (TOKEN-PTR).
           ADD 1 TO IN-PTR.
           IF IN-PTR IS NOT GREATER THAN 72
                MOVE IN-COL (IN-PTR) TO IN-CHR
                IF IN-CHR IS NOT EQUAL TO ´(´
                     GO TO EXT-CHR-LOOP2
                ELSE
                IF TOKEN-CHR (TOKEN-PTR) IS EQUAL TO ´(´
                     GO TO EXT-CHR-LOOP2.
           MOVE SPACE TO IN-CHR.
       EXT-TOKEN-END.
           IF IN-CHR IS EQUAL TO ´.´
           AND IN-PTR IS NOT GREATER THAN 72
                IF IN-COL (IN-PTR) IS NUMERIC
                OR IN-COL (IN-PTR)
                IS EQUAL TO ´E´
                     ADD 1 TO TOKEN-PTR
                     MOVE IN-CHR TO TOKEN-CHR (TOKEN-PTR)
```

```
                    MOVE IN-COL (IN-PTR) TO IN-CHR
                    GO TO EXT-CHR-LOOP2,
                ELSE ADD 1 TO NSENT.
            MOVE IN-CHR TO TOKEN-PUNC.
            MOVE TOKEN-PTR TO TOKEN-LEN.
            ADD TOKEN-LEN 1 GIVING TOKEN-SIZE.
            IF TOKEN-PUNC IS NOT EQUAL TO SPACE
                ADD 1 TO TOKEN-SIZE.
            GO TO EXT-EXIT.
        EXT-N-N-LITERAL.
            ADD 1 TO TOKEN-PTR.
            MOVE IN-CHR TO TOKEN-CHR (TOKEN-PTR).
            ADD 1 TO IN-PTR.
            IF IN-PTR IS GREATER THAN 72
                GO TO EXT-N-N-LIT-END.
            IF IN-CHR EQUALS QUOTE
                IF IN-COL (IN-PTR) EQUALS QUOTE
                    ADD 1 TO IN-PTR
                ELSE IF TOKEN-PTR GREATER
                    THAN 1 GO TO EXT-N-N-LIT-END.
            MOVE IN-COL (IN-PTR) TO IN-CHR.
            GO TO EXT-N-N-LITERAL.
        EXT-N-N-LIT-END.
            MOVE SPACE TO TOKEN-PUNC.
            MOVE TOKEN-PTR TO TOKEN-LEN.
            ADD TOKEN-LEN 1 GIVING TOKEN-SIZE.
            PERFORM NEXIT.
            GO TO EXT-EXT.
        EXT-EXIT.
            PERFORM CLASSIFY-TOKEN.
        EXT-EXT.
            EXIT.
        CLASSIFY-TOKEN SECTION.
            IF TOKEN-LEN IS GREATER THAN 15
                GO TO NEXIT
            ELSE GO TO L1, L2, L3, L4, L5, L6, L7, L8, L9,
                L10, L11, L12, L13, L14, L15
                DEPENDING ON TOKEN-LEN.
        L1.
            SEARCH ALL STR1, AT END MOVE 0 TO TABINX,
            GO TO NEXIT;  WHEN K1 (TABINX) = T1
            ADD 1 TO KW1 (TABINX),
            ADD 1 TO NRESW, GO TO CLASTOK-EXIT.
        L2.
            SEARCH ALL STR2, AT END MOVE 0 TO TABINX,
            GO TO NEXIT;  WHEN K2 (TABINX) = T2
            ADD 1 TO KW2 (TABINX),
            ADD 1 TO NRESW, GO TO CLASTOK-EXIT.
        L3.
            SEARCH ALL STR3, AT END MOVE 0 TO TABINX,
            GO TO NEXIT;  WHEN K3 (TABINX) = T3
            ADD 1 TO KW3 (TABINX),
```

```
         ADD 1 TO NRESW, GO TO CLASTOK-EXIT.
L4.
         SEARCH ALL STR4, AT END MOVE Ø TO TABINX,
         GO TO NEXIT; WHEN K4 (TABINX) = T4
         ADD 1 TO KW4 (TABINX),
         ADD 1 TO NRESW, GO TO CLASTOK-EXIT.
L5.
         SEARCH ALL STR5, AT END MOVE Ø TO TABINX,
         GO TO NEXIT; WHEN K5 (TABINX) = T5
         ADD 1 TO KW5 (TABINX),
         ADD 1 TO NRESW, GO TO CLASTOK-EXIT.
L6.

         SEARCH ALL STR6, AT END MOVE Ø TO TABINX,
         GO TO NEXIT; WHEN K6 (TABINX) = T6
         ADD 1 TO KW6 (TABINX),
         ADD 1 TO NRESW, GO TO CLASTOK-EXIT.
L7.
         SEARCH ALL STR7, AT END MOVE Ø TO TABINX,
         GO TO NEXIT; WHEN K7 (TABINX) = T7
         ADD 1 TO KW7 (TABINX),
         ADD 1 TO NRESW, GO TO CLASTOK-EXIT.
L8.
         SEARCH ALL STR8, AT END MOVE Ø TO TABINX,
         GO TO NEXIT; WHEN K8 (TABINX) = T8
         ADD 1 TO KW8 (TABINX),
         ADD 1 TO NRESW, GO TO CLASTOK-EXIT.
L9.
         SEARCH ALL STR9, AT END MOVE Ø TO TABINX,
         GO TO NEXIT; WHEN K9 (TABINX) = T9
         ADD 1 TO KW9 (TABINX),
         ADD 1 TO NRESW, GO TO CLASTOK-EXIT.
L1Ø.
         SEARCH ALL STR1Ø, AT END MOVE Ø TO TABINX,
         GO TO NEXIT; WHEN K1Ø (TABINX) = T1Ø
         ADD 1 TO KW1Ø (TABINX),
         ADD 1 TO NRESW, GO TO CLASTOK-EXIT.
L11.
         SEARCH ALL STR11, AT END MOVE Ø TO TABINX,
         GO TO NEXIT; WHEN K11 (TABINX) = T11
         ADD 1 TO KW11 (TABINX),
         ADD 1 TO NRESW, GO TO CLASTOK-EXIT.
L12.
         SEARCH ALL STR12, AT END MOVE Ø TO TABINX,
         GO TO NEXIT; WHEN K12 (TABINX) = T12
         ADD 1 TO KW12 (TABINX),
         ADD 1 TO NRESW, GO TO CLASTOK-EXIT.
L13.
         SEARCH ALL STR13, AT END MOVE Ø TO TABINX,
         GO TO NEXIT; WHEN K13 (TABINX) = T13
         ADD 1 TO KW13 (TABINX),
         ADD 1 TO NRESW, GO TO CLASTOK-EXIT.
```

```
L14.
        SEARCH ALL STR14, AT END MOVE 0 TO TABINX,
        GO TO NEXIT; WHEN K14 (TABINX) = T14
        ADD 1 TO KW14 (TABINX),
        ADD 1 TO NRESW, GO TO CLASTOK-EXIT.
L15.
        SEARCH ALL STR15, AT END MOVE 0 TO TABINX,
        GO TO NEXIT; WHEN K15 (TABINX) = T15
        ADD 1 TO KW15 (TABINX),
        GO TO CLASTOK-EXIT.
NEXIT.
        IF T1 > "@" AND T1 < "["
        ADD 1 TO NNRESW,
        ADD TOKEN-LEN TO SUMNRL,
        MULTIPLY TOKEN-LEN BY TOKEN-LEN GIVING TEMPR,
        ADD TEMPR TO SUMNRL2,
        MOVE T15 TO TTOKK,
        MOVE CARDCT TO TLEN,
        WRITE T-REC.
CLASTOK-EXIT.
        EXIT.
PRETTY-PRINT SECTION.
P1.
        IF KW1 (I) > 0 DISPLAY STR1 (I),"....",KW1 (I).
P2.
        IF KW2 (I) > 0 DISPLAY STR2 (I),"....",KW2 (I).
P3.
        IF KW3 (I) > 0 DISPLAY STR3 (I),"....",KW3 (I).
P4.
        IF KW4 (I) > 0 DISPLAY STR4 (I),"....",KW4 (I).
P5.
        IF KW5 (I) > 0 DISPLAY STR5 (I),"....",KW5 (I).
P6.
        IF KW6 (I) > 0 DISPLAY STR6 (I),"....",KW6 (I).
P7.
        IF KW7 (I) > 0 DISPLAY STR7 (I),"....",KW7 (I).
P8.
        IF KW8 (I) > 0 DISPLAY STR8 (I),"....",KW8 (I).
P9.
        IF KW9 (I) > 0 DISPLAY STR9 (I),"....",KW9 (I).
P10.
        IF KW10 (I) > 0 DISPLAY STR10 (I),"....",KW10 (I).
P11.
        IF KW11 (I) > 0 DISPLAY STR11 (I),"....",KW11 (I).
P12.
        IF KW12 (I) > 0 DISPLAY STR12 (I),"....",KW12 (I).
P13.
        IF KW13 (I) > 0 DISPLAY STR13 (I),"....",KW13 (I).
P14.
        IF KW14 (I) > 0 DISPLAY STR14 (I),"....",KW14 (I).
P15.
        IF KW15 (I) > 0 DISPLAY STR15 (I),"....",KW15 (I).
```

```
PP.
        DISPLAY O-S (I), O-SM (I).


XREF SECTION.
        CLOSE NKEYS,
        SORT SORT-FILE ASCENDING KEY A,B
        USING NKEYS GIVING NKEYS,
        OPEN INPUT NKEYS.
        MOVE 0 TO PCTR.
        MOVE SPACES TO P-TOK2.
        MOVE SPACES TO L.
XR.
        READ NKEYS INTO T-REC AT END
                IF PCTR > 0 DISPLAY PLINE, GO TO N-TERM
                ELSE GO TO N-TERM.
        IF TTOKK = P-TOK2
                IF PCTR = 9
                DISPLAY PLINE,
                MOVE SPACES TO L,
                MOVE 1 TO PCTR,
                MOVE TLEN TO L-FIELD (PCTR),
                 ELSE ADD 1 TO PCTR,
                MOVE TLEN TO L-FIELD (PCTR)
        ELSE
                DISPLAY PLINE,
                MOVE SPACES TO L,
                MOVE TTOKK TO P-TOK,
                MOVE TTOKK TO P-TOK2,
                MOVE 1 TO PCTR,
                MOVE TLEN TO L-FIELD (PCTR).
        GO TO XR.
N-TERM.
        DISPLAY "*** NORMAL TERMINATION ***".
        STOP RUN.
```

NBS-114A (REV. 7-73)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBS SP 500-26	2. Gov't Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE

COBOL Instrumentation and Debugging: A Case Study

5. Publication Date

January 1978

6. Performing Organization Code

7. AUTHOR(S) Gordon Lyon

8. Performing Organ. Report No.

9. PERFORMING ORGANIZATION NAME AND ADDRESS

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

10. Project/Task/Work Unit No.

11. Contract/Grant No.

12. Sponsoring Organization Name and Complete Address (*Street, City, State, ZIP*)

Same as 9.

13. Type of Report & Period Covered

Final

14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 77-608347

16. ABSTRACT (*A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.*)

Traditionally COBOL has been run in batch execution and debugged through abnormal terminations; indeed, whole books have been devoted to deciphering COBOL core dumps. With modern equipment there is little excuse for writing COBOL source code this way. A carefully chosen example (a COBOL program to tally reserved words and cross-reference other tokens) is used as a vehicle for an examination of a contemporary COBOL coding experience. COBOL can be written quite fast interactively, and COBOL programs can be easily tuned if good timing facilities are available. Strengths and limitations of a current debugging package are highlighted in the case study.

17. KEY WORDS (*six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons*)

COBOL; execution monitoring; symbolic debugging; timesharing.

18. AVAILABILITY [X] Unlimited

[ ] For Official Distribution. Do Not Release to NTIS

[X] Order From Sup. of Doc., U.S. Government Printing Office Washington, D.C. 20402, SD Cat. No. C13.10:500-26

[ ] Order From National Technical Information Service (NTIS) Springfield, Virginia 22151

19. SECURITY CLASS (THIS REPORT)	21. NO. OF PAGES
UNCLASSIFIED	33
20. SECURITY CLASS (THIS PAGE)	22. Price
UNCLASSIFIED	

USCOMM-DC 29042-P74

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents,
Government Printing Office,
Washington, D. C. 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Bureau of Standards Special Publication 500-.

Name ____

Company ____

Address ____

City ____________________ State ______________ Zip Code __________

(Notification key N-503)

NBS TECHNICAL PUBLICATIONS

PERIODICALS

JOURNAL OF RESEARCH—The Journal of Research of the National Bureau of Standards reports NBS research and development in those disciplines of the physical and engineering sciences in which the Bureau is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology, and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Bureau's technical and scientific programs. As a special service to subscribers each issue contains complete citations to all recent NBS publications in NBS and non-NBS media. Issued six times a year. Annual subscription: domestic $17.00; foreign $21.25. Single copy, $3.00 domestic; $3.75 foreign.

Note: The Journal was formerly published in two sections: Section A "Physics and Chemistry" and Section B "Mathematical Sciences."

DIMENSIONS/NBS

This monthly magazine is published to inform scientists, engineers, businessmen, industry, teachers, students, and consumers of the latest advances in science and technology, with primary emphasis on the work at NBS. The magazine highlights and reviews such issues as energy research, fire protection, building technology, metric conversion, pollution abatement, health and safety, and consumer product performance. In addition, it reports the results of Bureau programs in measurement standards and techniques, properties of matter and materials, engineering standards and services, instrumentation, and automatic data processing.

Annual subscription: Domestic, $12.50; Foreign $15.65.

NONPERIODICALS

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NBS, NBS annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a world-wide program coordinated by NBS. Program under authority of National Standard Data Act (Public Law 90-396).

NOTE: At present the principal publication outlet for these data is the Journal of Physical and Chemical Reference Data (JPCRD) published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements available from ACS, 1155 Sixteenth St. N.W., Wash., D.C. 20056.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The purpose of the standards is to establish nationally recognized requirements for products, and to provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order **above** *NBS publications from: Superintendent of Documents, Government Printing Office, Washington, D.C. 20402.*

Order **following** *NBS publications—NBSIR's and FIPS from the National Technical Information Services, Springfield, Va. 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. Register serves as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Services (Springfield, Va. 22161) in paper copy or microfiche form.

BIBLIOGRAPHIC SUBSCRIPTION SERVICES

The following current-awareness and literature-survey bibliographies are issued periodically by the Bureau:

Cryogenic Data Center Current Awareness Service. A literature survey issued biweekly. Annual subscription: Domestic, $25.00; Foreign, $30.00.

Liquified Natural Gas. A literature survey issued quarterly. Annual subscription: $20.00.

Superconducting Devices and Materials. A literature survey issued quarterly. Annual subscription: $30.00. Send subscription orders and remittances for the preceding bibliographic services to National Bureau of Standards, Cryogenic Data Center (275.02) Boulder, Colorado 80302.

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Washington, D.C. 20234

OFFICIAL BUSINESS

Penalty for Private Use, $300

POSTAGE AND FEES PAID
U.S. DEPARTMENT OF COMMERCE
COM-215

U.S.MAIL

SPECIAL FOURTH-CLASS RATE
BOOK