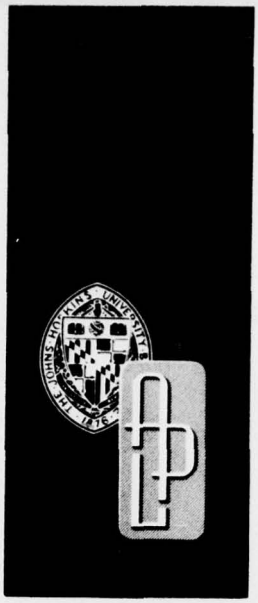
AD-A040 929 JOHNS HOPKINS UNIV LAUREL MD APPLIED PHYSICS LAB F/G 9/2
A CASE STUDY OF A PROGRAM LIBRARY TECHNIQUE.(U)
MAY 77 P R POPICK
UNCLASSIFIED APL/JAU-TG-1310 N00017-72-C-4401
NL

1 OF 1
ADA040929

END

DATE FILMED
7-77

APL/JHU
TG 1310
MAY 1977
Copy No. 1



Technical Memorandum

A CASE STUDY OF A PROGRAM LIBRARY TECHNIQUE

by P. R. POPICK

THE JOHNS HOPKINS UNIVERSITY • APPLIED PHYSICS LABORATORY

PLEASE FOLD BACK IF NOT NEEDED FOR BIBLIOGRAPHIC PURPOSES

REPORT DOCUMENTATION PAGE

1. REPORT NUMBER	2. GOVT ACCESSION NO	3. RECIPIENT'S CATALOG NUMBER
TG 1310		

4. TITLE (*and Subtitle*)	5. TYPE OF REPORT & PERIOD COVERED
A CASE STUDY OF A PROGRAM LIBRARY TECHNIQUE	Technical Report
	6. PERFORMING ORG. REPORT NUMBER TG 1310

7. AUTHOR(s)	8. CONTRACT OR GRANT NUMBER(s)
P. R. Popick	N00017-72-C-4401

9. PERFORMING ORGANIZATION NAME & ADDRESS	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
The Johns Hopkins University Applied Physics Laboratory Johns Hopkins Road Laurel, MD 20810	B4

11. CONTROLLING OFFICE NAME & ADDRESS	12. REPORT DATE
Naval Sea Systems Command SEA-652 Washington, DC 20362	May 1977
	13. NUMBER OF PAGES 21

14. MONITORING AGENCY NAME & ADDRESS	15. SECURITY CLASS. (*of this report*)
Naval Plant Representative Office Johns Hopkins Road Laurel, MD 20810	Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE NA

16. DISTRIBUTION STATEMENT (*of this Report*)

Approved for public release; distribution is unlimited.

17. DISTRIBUTION STATEMENT (*of the abstract entered in Block 20, if different from Report*)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (*Continue on reverse side if necessary and identify by block number*)

Configuration management
Program library
Software engineering
Structured programming

20. ABSTRACT (*Continue on reverse side if necessary and identify by block number*)

Program libraries have been used in the past in conjunction with structured programming, but very little attention has been given to the advantages of using a program library as a separate technique. As a separate technique, the program library increases project visibility and has the added advantage of being easily implemented with no retraining of the technical staff. The program library technique is outlined in this report and details of an implementation on a medium-sized project are given. This case study can serve as a guideline for an implementation of a program library on other projects. Many small-to-medium scale projects, that have heretofore ignored structured programming techniques because of personnel retraining costs and delay of implementation, may rapidly implement a program library to increase productivity, improve configuration management, and increase project visibility.

DD FORM 1 JAN 73 1473

APL/JHU
TG 1310
MAY 1977
Copy No.

Technical Memorandum

A CASE STUDY OF A PROGRAM LIBRARY TECHNIQUE

by P. R. POPICK

THE JOHNS HOPKINS UNIVERSITY ■ APPLIED PHYSICS LABORATORY
Johns Hopkins Road, Laurel, Maryland 20810
Operating under Contract N00017-72-C-4401 with the Department of the Navy

ABSTRACT

Program libraries have been used in the past in conjunction with structured programming, but very little attention has been given to the advantages of using a program library as a separate technique. As a separate technique, the program library increases project visibility and has the added advantage of being easily implemented with no retraining of the technical staff. The program library technique is outlined in this report and details of an implementation on a medium sized project are given. This case study can serve as a guideline for an implementation of a program library on other projects. Many small-to-medium scale projects, that have heretofore ignored structured programming techniques because of personnel retraining costs and delay of implementation, may rapidly implement a program library to increase productivity, improve configuration management, and increase project visibility.

CONTENTS

ILLUSTRATIONS

1. INTRODUCTION

In recent years the concept of a program library has been developed in conjunction with other structured programming techniques (Ref. 1) in order to facilitate the systematic development of computer software. In a general sense, structured programming refers to the systematic development of software and includes other techniques such as "top-down design," "chief programmer teams," and "structured walk-throughs" (Refs. 2 and 3).

Although there have been several articles published in the literature on using program libraries with other structured programming techniques, very little consideration has been given to the use of a program library as an independent technique that is easily and quickly implemented since no special education is required for the technical staff. Other structured programming techniques require a fair amount of training.

As an independent technique, the program library can make improvements in areas of software development: (a) in the productivity of the technical personnel because support personnel are used to perform routine functions, (b) in program configuration control and management visibility, and (c) in training support personnel who can perform routine functions (such as keypunching, record keeping, source updates, and submittal and pickup of batch computer jobs).

In Section 2 a case study is presented of a program library implemented for a medium-sized software project. Details are included of some software development problems that were eliminated by the use of the program library. Section 2 could serve as a guideline for the implementation of a program library for other software projects. Actually a program library is nothing more than a collection of good practices, some of which have been used on various projects for years. The strength of the program library technique lies in the collection of these good practices and the organization of them into formal procedures.

Ref. 1. O. J. Dahl, E. W. Dijkstra, and C. A. R. Hoard, Structured Programming, Academic Press, London, England, 1972.

Ref. 2. F. T. Baker and H. D. Mills, "Chief Programmer Teams," Datamation, December 1973, pp. 58-61.

Ref. 3. R. C. McHenry, "Management Concepts for Top Down Structured Programming," Internal IBM memorandum FSC-73-001. Federal System Division, Gaithersburg, MD.

The program library discussed here was initiated for an on-going project that involved ten program designers. In the project, a military mini-computer was used that had a limited amount of support software and required cards or magnetic tape as the input medium. Prior to the introduction of the program library, each programmer worked independently and was responsible for updating source code, submittal and pickup of batch computer jobs, and managing his own test configuration. A standard keypunch service was available but, for expediency, most program designers keypunched their own update cards. The program library was designed to assume responsibility for these routine functions. Consequently the program designer could spend more time on software design and development. The program library was gradually phased in over a three-month period so that the program designers and support personnel could become comfortable in their new roles.

2. PROGRAM LIBRARY OPERATION

The program library performs routine functions connected with software development. These functions include: source updating, assemblies, linkage editing, job pickup and delivery, and configuration management. When a program library is in use, the program designer is able to submit hand-marked corrections on assembly listings and receive in return an updated executable test-module tape. Library personnel do the keypunching, set up the appropriate control decks, run or sumbit batch jobs, verify results, and keep records on the latest module revision and test environment. Turnaround for jobs should average less than one day; large work requests are routed to normal keypunch services to free library personnel for other tasks. Figure 1 is an overview of the update and assembly function.

The following example illustrates how the program library and the programmer interact.

A programmer marks changes to his source code on an assembly listing; large code insertions are indicated on coding sheets. Then the programmer submits the listing, coding sheet, and work request to the program library. The librarian makes the indicated changes, updates the source, and verifies the changes. The new source is given a new revision number, and an entry in the program status book is made that indicates the new module revision, date, and person responsible. The new source and relocatable object are stored on tape, and the tape is stored in the tape library. The new listing is returned to the program designer.

If, following several iterations through testing and updating of the module, the programmer decides his module has been completely tested, the librarian places the new module into a standard test environment for access by other module testers. All corrections to the test environment are entered into the program test environment relocatable object and creates an executable bootstrap tape. The programmer then performs "hands-on" checking of the new module. As errors are encountered, the program designer may request further corrections, in which case the above procedure is repeated as many times as is necessary.

If following several iterations through testing and updating of the module, the programmer decides his module has been completely tested, the librarian places the new module into a standard test environment for access by other module testers. All corrections to the test environment are entered into the program designer's

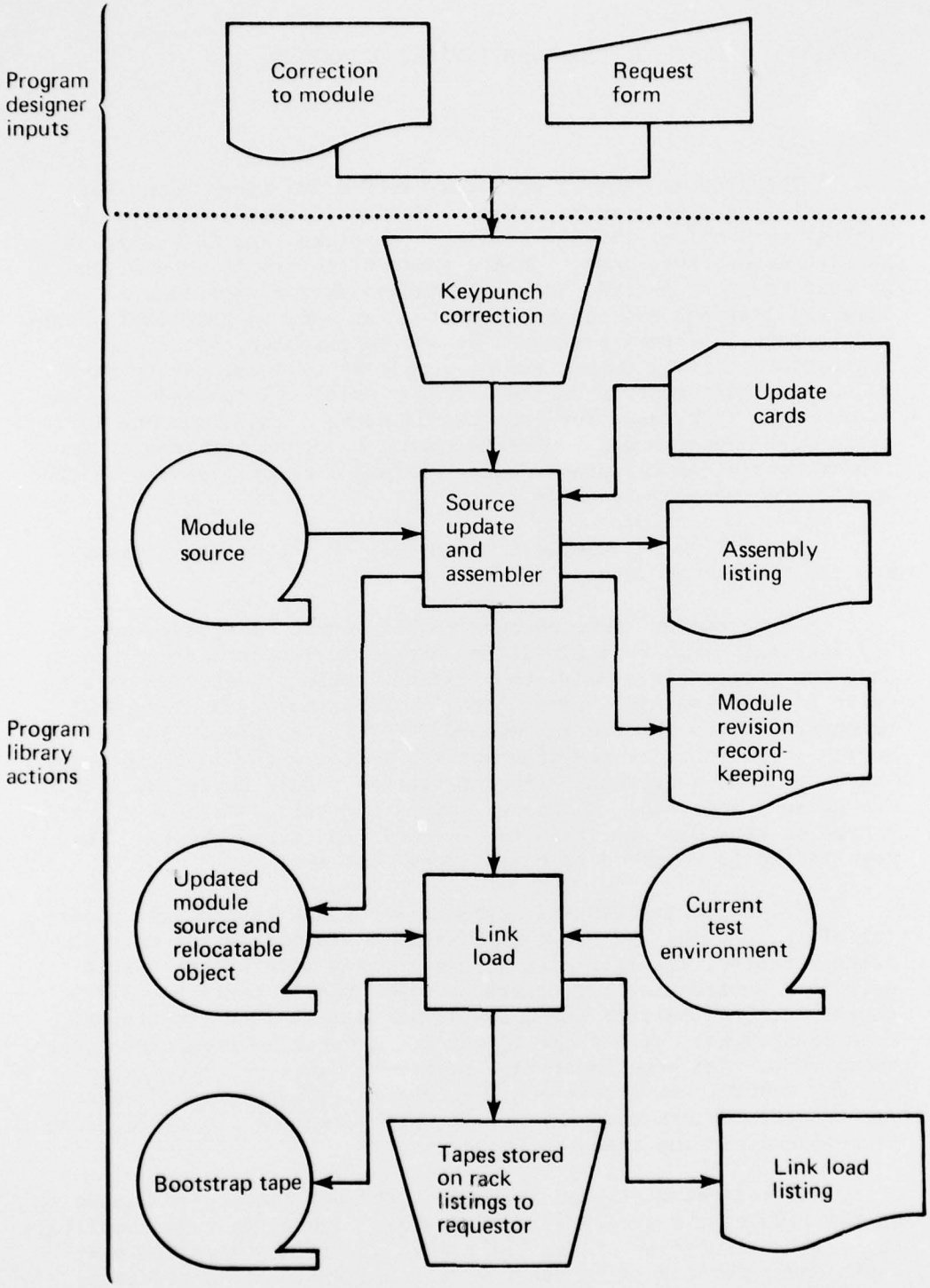


Fig. 1 Creation of a Test Bootstrap Tape

designer's test tape through the program library. An example of how the program designer interfaces with the program library is shown in Fig. 2.

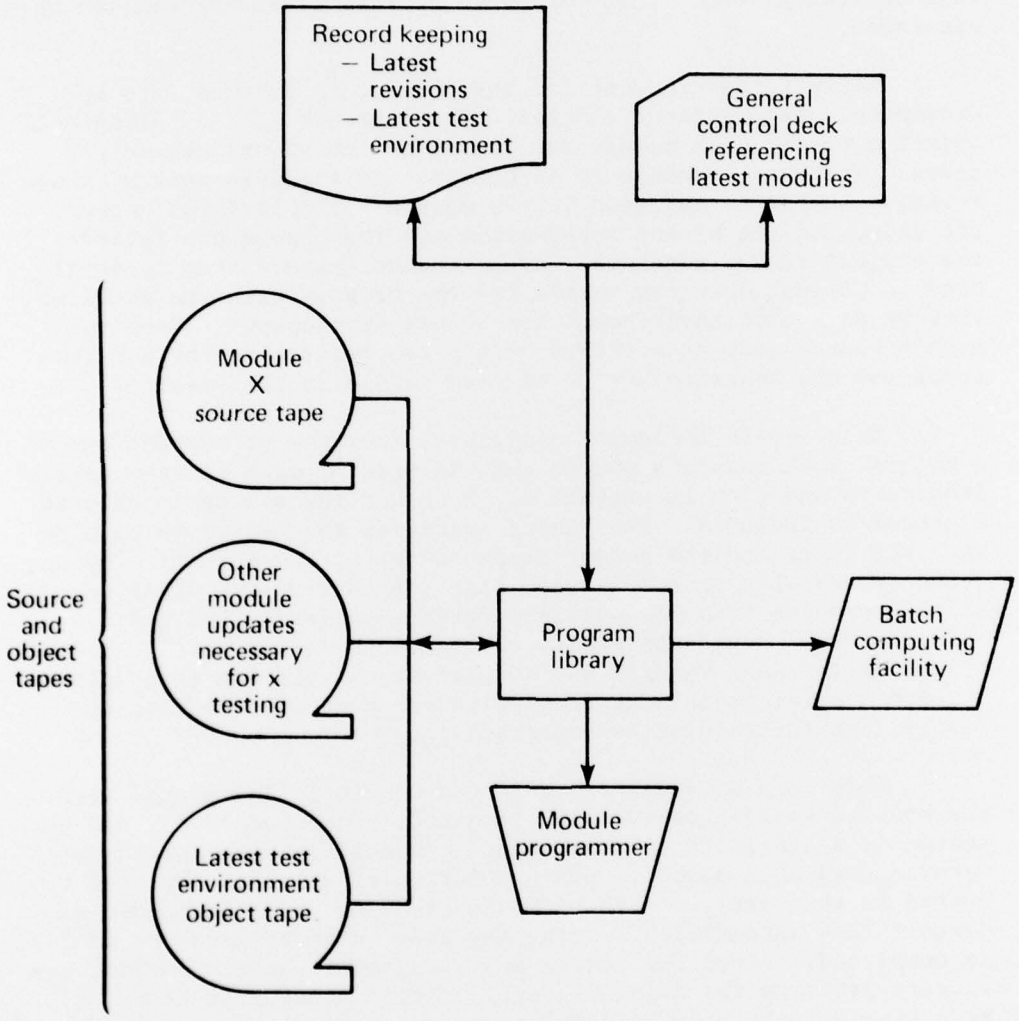


Fig. 2 Configuration Control with Program Library

3. CONFIGURATION MANAGEMENT

One especially useful aspect of the program library that bears special attention is the configuration management system created by the introduction of the program library. In this section the configuration management system is briefly described so that several pitfalls that have been avoided by the system can be discussed.

Software development for the project is divided into approximately ten distinct modules. The source code and relocatable object code for each module are stored on individual magnetic tapes. Whenever a change or an addition to a module occurs, a new revision number is assigned to the module. The revision number, the date, and the person responsible for the change are recorded in the project status notebook. Since the software system is developed in phases, each new version of the program uses the previous version as a test environment for module development. Each version's source code is archived with notes describing the new features and the revision number of each module in the version.

In order to implement additional features or corrections to a module, each module's source code is updated with the new source statements and then is assembled. Whenever the source is updated, a header is included. The header specifies the new revision number, the date, and the person responsible for the change. The updated relocatable object is then link loaded with the other unchanged modules from the previous version to verify the updates. This procedure continues separately for each module until all features for the next version are completed. As testing is completed on each module, it is integrated with other updated modules to verify that the modules can interact properly.

Each time an updated and tested module is integrated with the program version the version is dated. Each module can now be tested in a static test environment by specifying the same dated version, and each module, upon completion of testing, may be intgrated in an orderly manner without affecting individual module tests. Thus integration testing may proceed as testing of modules is completed. Since the module and integration tests overlap, the library performs the rigorous configuration management tasks to keep straight the many configurations necessary.

When module testing has been completed, the new version is released, and the modules are archived in several ways to ensure against loss. The changes and additions for the newest version

are written on to the archive tape and on the test-bed tapes. Further notes, backups, and listings are maintained in the program library room. Additionally all changes to a module, from one version to another, are retained so that any module revision can be recreated if necessary.

4. CONFIGURATION MANAGEMENT WITHOUT PROGRAM LIBRARY

The configuration management system described in Section 3 is very similar to the system used prior to the introduction of the program library. However, there were several important differences that allow the program library to be a great deal more effective in configuration management.

First the program library serves as the focal point for all record keeping and status information, thus eliminating the problem of losing updates to modules when two people change the same module. In the old system the programmer picked up the latest revision of a module from the tape rack, updated the module, and restored the old tape to the tape rack. The programmer then took the updated tape, used it, and eventually returned it on the tape rack. If, during the time the updated tape was being used, another programmer made an update to the same module using the same latest revision of the tape, two revisions of a module would be in use. Both modules would be missing an update. If the second change was undetected, confusing results would be obtained. With the program library, all work is funneled through a central facility with each module revision entered in the log. Thus, when the second programmer submits an update for the module, library personnel can refer to the log and the latest revision would automatically be used.

The second improvement provided to configuration management by the program library occurs in the maintenance of a stable test environment. With the old system, each programmer maintained his own test environment by taking the standard versions of the other modules plus any specially updated modules necessary for his testing and then would work independently until testing was complete. However, when the programmer corrects the errors he has found in his test environment, he does not necessarily communicate these changes to the other module testers. As a result, time can be wasted in detecting the same problem in each programmer's special test environment. With the program library the basic testing environment is similar for all programmers. Thus when an error in the test environment is detected and corrected, it will be incorporated promptly into each programmer's test environment.

The third improvement effected by the program library is the rigorous archive system. Several duplicate or backup tapes are made of previous source modules and test environments. Each backup tape is carefully labeled and entered in the module and test environment revision log. Prior to the installation of these

procedures, the responsibility was left to individual programmers, who all had other responsibilities and were unable to make backup versions regularly. Consequently, when previous versions were needed, no source or object code was available. Figure 3 illustrates the program designer configuration access without the program library. Most problems occurred because of the concurrency of operation with this system.

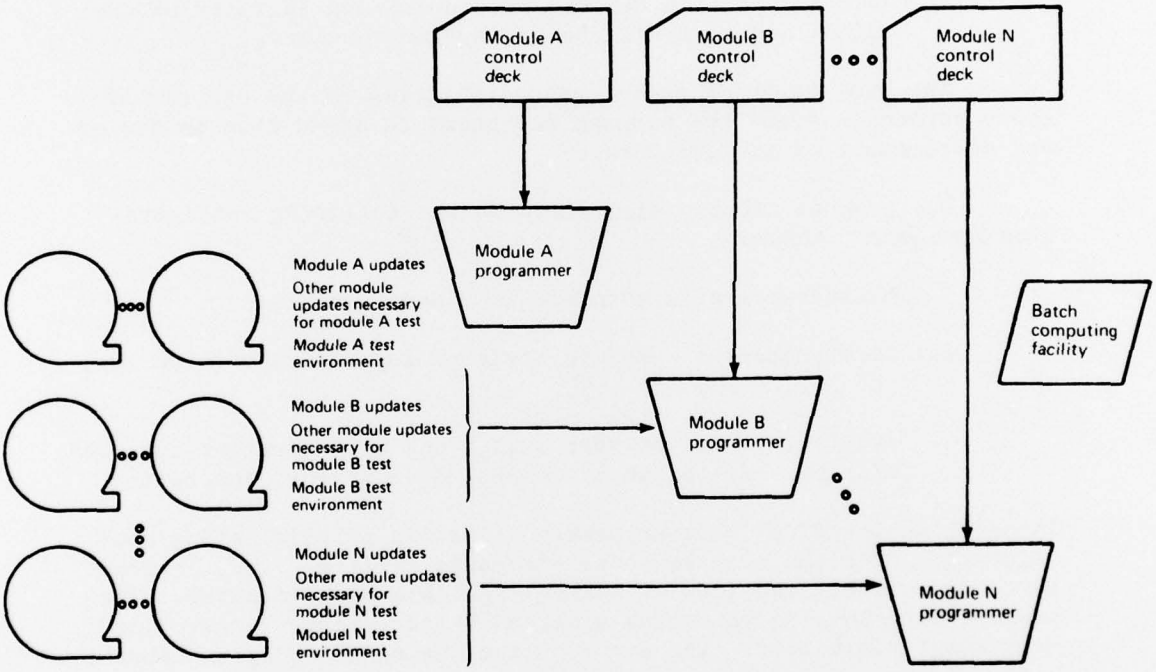


Fig. 3 Configuration Access without Program Library

5. CONCLUSIONS

Program library personnel perform the following tasks that were previously performed by the program designers:

1. Keypunching and module updating;
2. Submittal and pickup of batch computer jobs;
3. Routine backup functions, archiving, and record keeping; and
4. Maintenance of a central record-keeping facility of the latest source revisions and system versions.

The assumption of these responsibilities by the program library personnel freed the program designers to spend time on design and development of the software.

The program library also performs the following configuration management tasks:

1. Maintenance of a standard test environment;
2. Maintenance of a module revision log and an archive log; and
3. Availability of current status and configuration information for each phase of design, development, and test.

These three functions were extremely effective in eliminating configuration problems such as those discussed earlier; i.e., improper test environments and loss of source updates were eliminated. The added record-keeping mechanism afforded better program control and management visibility. The net result of using the program library was:

1. Increased productivity of technical personnel;
2. Better organized project development; and
3. Higher quality software.

The final improvement accomplished by the program library is a less tangible result, but nontheless perhaps the most important. The program library provides a central point for communication with all members of the project. This additional communication

seems to pull the project members closer together, resulting in project personnel functioning as a well-coordinated team rather than as individuals. The program library proved to be easily implemented with little or no retraining of technical personnel and produced the desired results in a very short time. As a result, the program library technique may prove to be especially useful for small- to medium-sized software projects that have limited budgets and resources to produce the above results.

REFERENCES

1. O. J. Dahl, E. W. Dijkstra, and C. A. R. Hoare, Structured Programming, Academic Press, London, England, 1972.

2. F. T. Baker and H. D. Mills, "Chief Programmer Teams," Datamation, December 1973, pp. 58-61.

3. R. C. McHenry, "Management Concepts for Top Down Structured Programming," Internal IBM memorandum FSC-73-001. Federal Systems Division, Gathersburg, MD.

INITIAL DISTRIBUTION EXTERNAL TO THE APPLIED PHYSICS LABORATORY*

The work reported in TG 1310 was done under Navy Contract N00017-72-C-4401. This work is related to Task B4, which is supported by NAVSEASYSCOM.

ORGANIZATION	LOCATION	ATTENTION	No. of Copies
DEPARTMENT OF DEFENSE			
DDC	Alexandria, VA		12
Department of the Navy			
Chief of Naval Material	Washington, DC	NMAT-09Y	1
NAVSEASYSCOM	Washington, DC	SEA-06C	1
		SEA-09G3	2
		SEA-652	1
		SEA-6542	1
NAVELEXSYSCOM	Washington, DC	PMS-403B	1
NAVPRO	Laurel, MD	NLEX 570	1
MISCELLANEOUS			
Bell Laboratories	Whippany, NJ	R. W. Held, 1E-321A	1
		P. R. Popick, 2D-306A	1
	Naperville, IL	M. B. Defrenza	1
IBM	Gathersburg, MD	C. V. Walker	1
	Manassas, VA	V. M. FeLarka	1

Requests for copies of this report from DoD activities and contractors should be directed to DDC, Cameron Station, Alexandria, Virginia 22314 using DDC Form 1 and, if necessary, DDC Form 55.

*Initial distribution of this document within the Applied Physics Laboratory has been made in accordance with a list on file in the APL Technical Publications Group.