C

Programming Support Library (PSL) Operations Guide

IBM

(9) Final rept.

30272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO. (19) (18) DOD/DE-81/016c	2.	3. Recipient's Accession No. AD-A107 251
4. Title and Subtitle (6) PROGRAMMING SUPPORT LIBRARY (PSL), Operations Manual, [redacted]			5. Report Date (11) May 1978
			6. (12) 27
7. Author(s)			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Federal Systems Division International Business Machines Corporation Gaithersburg, Maryland			10. Project/Task/Work Unit No. 11. Contract(C) or Grant(G) No. (15) (C) F30602-77-C-0249 (G)
12. Sponsoring Organization Name and Address Rome Air Development Center RADC/COEE Griffiss Air Force Base, NY 13441			13. Type of Report & Period Covered 14.

15. Supplementary Notes

(21) For magnetic tape, see AD-A107248.
For disc pack AD-A107 252.

The source agency has restricted sales of this item to Federal, state and local governments.

16. Abstract (Limit: 200 words)

The Programming Support Library (PSL) is a software system which provides the tools to organize, implement, and control computer program development. This involves the support of the actual programming process and also the support of the management process. The PSL is designed to support Top Down Design and Structured Programming (TDDSP).

The objective of the Operations Manual for the PSL is to provide computer control and computer operator personnel with a detailed operational description of the system and its associated environment.

17. Document Analysis a. Descriptors

Software Configuration Control
Structured Programming Support Tool
Software Development Support

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group 174950

18. Availability Statement:	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages
	20. Security Class (This Page) UNCLASSIFIED	22. Price

(See ANSI-Z39.18) See Instructions on Reverse OPTIONAL FORM 272 (4-77)

IBM-G-4

DMA

Programming Support Library (PSL)

Operations Manual (FINAL)

Submitted to:

Rome Air Development Center

Griffiss Air Force Base, New York

May 1978

Under Contract F30602-77-C-0249

Federal Systems Division
INTERNATIONAL BUSINESS MACHINES CORPORATION
Gaithersburg, Maryland

TABLE OF CONTENTS

LIST OF FIGURES

SECTION 1. GENERAL DESCRIPTION

1.1 Purpose of Computer Operation Manual

The objective of the Computer Operation Manual for the Programming Support Library System, developed under contract number F30602-77-C-0249, is to provide computer control and computer operator personnel with a detailed operational description of the system and its associated environment.

The manual is organized into four sections as follows:

a. SECTION 1 - GENERAL DESCRIPTION. This section describes the purpose and use of the PSL system; the operation of the system, in regards to how the various operations are related; and the associated input data and output information for each operation.

b. SECTION 2 - SYSTEM CONTROL. This section presents information on how the operations and environment associated with the system are controlled. It also describes the responsibilities and interactions of control personnel in the scheduling of operations; assignment of equipment; and the management of input data, output data and other output information in an orderly way.

c. SECTION 3 - OPERATING PROCEDURES. This section describes the procedures and material required to perform each of the individual operations of the PSL. This description will be illustrated by defining a hypothetical project and itemizing the logical sequence of actions to be taken.

d. SECTION 4 - NON-ROUTINE OPERATIONS. This section provides control information and operator procedures to cover non-routine PSL operations.

1.2 System Application

The DMA PSL is a comprehensive system software package which supports the growth and maintenance of structured programming projects in a top-down development environment. The system provides:

a. A framework for the organization of a project.

b. Simple functional statements to interface between the programmer and the machine.

c. Structural and statistical reports for control of the development process and for communication between programmers.

The DMA PSL system provides special structured programming support for the Structured COBOL and Structured FORTRAN languages. However, unstructured program may also be stored and maintained under the system.

1.3 System Operation

The PSL is primarily a self-contained software system with most functions being performed by programs contained within the system. The PSL does however, interface with several external support programs as follows:

a. UNIVAC 1108 Executive System. The execution of the PSL programs and the processing of Job Control statements stored in the library are controlled by the Executive System.

b. File Control Processor. The reading and writing of library data sections on the direct access storage device is supported by the File Control Processor.

c. Language Processors. Compiles the program source language statements.

d. Collector. The collector is designed to provide the user with the means of collecting and linking relocatable subprograms to produce an absolute program in a form ready for execution under control of the Executive System.

The PSL system in the batch mode is invoked by an ADD statement and is directed to perform specific functions by PSL Function cards. The general functional capabilities available under the PSL are:

a. Initialize a project.

b. Create sections in a library.

c. Add, change, move, replace or purge a unit of code.

d. Print, punch or write (to tape) a unit of code.

e. Print an index listing.

f. Print the top-down structure of a program.

g. Compile, link, execute.

h. Delete a section, a library or a project.

i. Backup a project, library or section.

j. Restore a project, library or section.

k. Collect and print management data.

l. Print text data.

m. Print by author or character string.

The PSL is used to maintain the current status of all code in machine readable form. The current status of the PSL includes, at the least, library sections which contain source and object code and control data. Other user generated data such as program design language statements, test data and textual data may also be maintained in the PSL. It is also used to maintain the current status and past history of all code generated in a project in a human readable form. For human readability, the various types of source data, test data and control data sections are represented by listings of their members. The object code is represented by the assembler/compiler listing produced when the object code was created.

A PSL library consists of one or more data sections. The defined sections are:

a. SOURCE - This section is used to store source statements. It serves as input to the precompilers during compilation of a program.

b. OBJECT - This section contains object module indexes and accounting records. The object code generated from a compile/assembly is located in the PROGRAM section. (See k.)

c. LINK - This section is used to store control statements for combining one or more object and load modules recorded in the OBJECT and LOAD sections into an executable load module. The resulting load module is recorded in the LOAD section and stored in the PROGRAM section.

d. LOAD - This section contains load module indexes and accounting records.

e. PDL - This section is used to store Program Design Language statements. Data from this section may be used for design or program documentation.

f. TEXT - This section is used to store any textual type information. Data from this file may be used for various types of program or system documentation.

g. JOB - This section is used to store job control statements which are used to control execution of programs being developed and other programs used by the programming project. Data from the section is used to build a job stream for input to the computer.

h. TEST - This section is used to store data to be used in testing programs under development or maintenance.

i. MGMT - This section is used to store management data to be used in producing management data reports.

j. USER - This section contains data generated by non-PSL functions.

k. PROGRAM - This section contains all object and load modules (relocatable and absolute elements).

The PSL system manipulates the data for each system appropriately for that type of section.

Data is stored in the various library sections in the form of units. A unit is composed of one or more lines of data (a line is equivalent to seventy-two characters of data plus sequence number) except in the case of object and load modules, text data, and when the data compaction option is being used. Compacted data consists of variable length records. Object and load modules are generated by system facilities and stored automatically. All other data units are user generated and stored as a result of a user PSL update run.

The Batch Control Function, BCTL, provides control for the PSL in a batch environment. Its primary functions are:

a. Process the input card.

b. Perform job request verification.

c. Call the program to be executed, passing control information entered via the control card to the program.

BCTL is loaded into core storage from the computer's system resident file and executed when a PSL job is initiated by a job control statement. As PSL control cards are read, BCTL processes the input card and passes control to one of the functional processors. When a functional processor completes its processing, control is returned to BCTL. The Batch Control Function continues to load control cards and execute functional processors until an end-of-file condition is encountered.

The BCTL Function uses the first parameter on the PSL control card to determine which functional processor to execute. It is the function of each processor to check for its required keyword parameters.

SECTION 2. SYSTEM CONTROL

2.1 Control Requirements

The PSL is a repository for the storing of data created during the development of computer programs. The data repository is in two forms: data stored in machine readable form accessible by the computer and identical data stored in hard copy form in project notebooks. The necessary computer and office procedures for manipulation of this data is also included as a part of the total PSL system. The purpose of the PSL is to support the program development process. This involves the support of the actual programming process and the management of the programming process. Participation by management, programmers and librarian is required to effectively utilize the capabilities of the total system.

The programming project manager is responsible for defining and monitoring a set of office procedures and computer procedures used to control the PSL library. Detailed instructions for their use and for preparing inputs and outputs must be provided.

The contents of the PSL are maintained by a programming librarian and determined by programmer. Except for normal housekeeping functions, clerical operations are to be carried out in the PSL only on the direct request of the programmers. Under such direction, the programmer librarian carries out any of the operations in the PSL without assistance or direct supervision. Directions from programmers to the programming librarian may come through marked up members, original code sheets or notations on directories. If a programming librarian is not used, an appropriate standard set of machine and office procedures should be employed by the programmers.

The management process consists of the following four basic functions:

a. Planning. The Function of determining the project objectives and the policies, programs, procedures and methods for achieving them. The planning must provide a framework for decision making.

b. Organizing. The Function of determining the activities required to achieve the objectives of a programming project, the departmentation of these activities and the assignment of authority and responsibility for their performance.

c. Control. The Function of assuring that the various components of a project are performing in accordance with the plan. Control is essentially the measurement and modification (if necessary) of component activities to assure the accomplishment of the overall plan.

d. Communications. The Function of transferring information among decision makers throughout the project.

The PSL may be used to support the management of the programming process by providing a means to:

- Collect and report management data related to program development.
- Control the integrity and security of the data stored in the PSL.
- Separate the clerical activity related to the programming process.

The management statistical data stored in the PSL should be maintained by a programming librarian under the supervision of project management. Except for normal housekeeping functions, clerical operations for the maintenance and reporting of management statistical data should be carried out in the PSL only on the direct request of managers or on a previously defined management approved request procedure.

2.2 Data Retention

An external library consisting of a set of current status notebooks, archives and run notebooks which reflect the contents of the internal PSL library and provide a history of the programming project should be maintained as an integral part of the total development process. For each internal file maintained by the PSL system there should be a corresponding external file in the form of printed listings stored in notebooks or binders.

Each of the external libraries should also contain a directory (list of unit names) and a control listing related to the physical attribute of the file. The contents are filed in alphabetical sequence by unit name with the directory and control listing preceding the unit listing.

Archives are maintained for each of the current status notebooks. These provide a history of the development process, a basic backup and management control of resources. Each archive is a chronological collection of old listings for a notebook.

A Housekeeping archive and a General archive are also maintained. The Housekeeping binder contains output from the PSL system maintenance functions (See Section 4) and is filed in chronological sequence. The General archive contains any output which is not filed in one of the other archives. It is also maintained in chronological order.

The length of time for which material is maintained in each archive is a project-dependent consideration.

Run notebooks contain the output from executions of system test runs. The run notebook provides a chronological history of the execution status of the modules it represents. The number of different run books kept is a project-dependent consideration.

SECTION 3. OPERATING PROCEDURES

3.1 Equipment Configuration

The PSL system operates on a Sperry Univac 1100 series computer under the EXEC 8 Operating System using a standard DMA software and hardware configuration. Libraries are maintained on direct access storage devices. The system utilizes the standard card reader and printer, one tape drive, and approximately 43K words of core.

The PSL system utilizes the File Control Processor to dynamically create, allocate, deallocate and purge file storage space. Interface with the File Control Processor is made through the Executive Request command using the Univac Assembly Program Language.

3.2 Input

The input to the PSL system consists of directives on cards (PSL Function cards) and data from the user's libraries. A PSL directive which is submitted as a batch-input data card via the card reader or as terminal input, contains the name of the function requested and, if appropriate, sets of keywords and value entries. The user's own program and data cards may be interspersed with the PSL Function cards.

The format of the input card is:

```
** function  keyword-parameter,keyword-parameter,...
```

where

**	- Present in positions 1 and 2 and followed by one or more blanks. These characters identify all PSL control cards.
function	- Must be present and followed by one or more blanks. This entry defines the particular PSL function or subfunction to be performed.
keyword-parameter	- Additional control information related to the function performed. Multiple keyword parameters are separated by commas and may occur in any order. The keyword parameters

are defined in the descriptions of the PSL function processors. The format of a keyword parameter is

Keyword=value

The entire KEYWORD need not be specified. Only those characters needed to distinguish between keywords are required.

If the keyword parameters will not fit on a single punched card, continuation cards are permitted. Continuation cards are identified as follows:

a. The control card preceding the continuation card must contain a comma as the last character on the card.

b. A continuation card must contain a ** in columns 1 and 2 followed by one or more blanks and the additional keyword parameters. The keyword parameters are formatted exactly as on the standard PSL control card.

When a required keyword parameter is missing or in error, the processing for that function is terminated with an error message. Normal processing is continued for any additional functions requested.

3.3 Output

The output of the PSL system consists mainly of user data which are stored according to data format in pre-defined files (PSL sections), and management data reports.

The Output Processing functions (Section 3.1.5, PSL Users Manual) provide the facility to output data stored in the PSL library. First, the PRINT INDEX Function provides a control listing which reports on the status of each section. Information related to physical storage including storage space allocated and used, type of storage unit, data identification and similar type information is included. Next, the PRINT SOURCE Function provides either a listing, tape or card deck for either a single unit or each unit in a section. The PRINT AUTHOR Function provides the capability to list the name of each unit in a section generated by a specific programmer. The PRINT

DOCUMENT Function prints document text incorporating card-image unit stored in PSL sections. The SCAN STRING Function provides the capability to scan for and list each appearance of a specified character string in a section.

In addition to the above functions, a unit listing is produced automatically each time a unit of code is either added or updated within the SOURCE, JOB, TEST, MANAGEMENT, TEXT and PDL sections. Units added to either the SOURCE or PDL sections will be listed with indentation, if appropriate.

Management data reports print the contents of one or more management data units in a pre-defined report format. Management data units may contain a combination of automatically collected data and manually input data combined according to the specifications in an associated user-defined format unit.

The Management Data Report Function allows the user to request the generation of reports containing management information on active programming projects or historical programming projects. All reports are produced upon request. The makeup of the selected set of reports, variances between data items (Exception Checking), and the reporting cycle are determined by the project personnel.

3.4 Procedures

The following is an example of how the PSL may be used to develop a segment of code by an individual programmer. These procedures may be extended to encompass the development of an entire system by many programmers.

A description of the operation (PSL Function) to be performed is given, followed by a sample of the input needed to implement each operation.

```
@ASG,A  DMA*PSL.
@USE    PSL.,DMA*PSL.

Initialize a PSL project and create two libraries, DEVEL (development) and INTEG
(integration), with all their required sections.


 @RUN   run-id,account-#,project-id
 @ADD,L   PSL.RUN
** INITIAL    PROJECT=5124SSSPSL                          INITIALIZE PROJECT
** CREATE     PROJECT=5124SSSPSL,LIBRARY=DEVEL,           {CREATE DEVELOPMENT
**                  SECTION=JOB                           {LIBRARY
** CREATE     SECTION=SOURCE,MGMTDATA=YES
** CREATE     SECTION=MGMT,FMS=(15,15)
                  .
                  .
                  .
** CREATE LIBR=INTEG,SECTION=JOB                          {CREATE  INTEGRATION
** CREATE      SECT=OBJECT                                {LIBRARY
                  .
                  .
                  .
 @END  PSL
 @XQT    PSL.BCTL
 @FIN
```

Plan management data report - Develop the structure of the report, identify data items to be recorded, initialize manual input items, and insert data item check(s).

```
@RUN    ----
@ASG,A   DMA*PSL.
@USE     PSL.,DMA*PSL.
@ADD,L   PSL.RUN
** PARAM  PROJECT=5124SSSPSL,LIBRARY=DEVEL
** MDPLAN   MOD=0                                          {GENERATE MANAGEMENT
** INSERT AFTER=0                                         {DATA REPORT FORMAT
  SYSTEM=TIPTOP-SYSTEM
    SUBSYSTEM=TIPTOP-SUBSYSTEM                             STRUCTURE OF  REPORT
      MODULE=TIPTOP
** MDFORMAT  LEVEL=SYSTEM,OP=ADD                          {DATA TO BE COLLECTED
 I010 RPT 48PROJ-DESCR    PROJECT DESCRIPTION             {IN REPORT
 I020     N06START-DATE   PROJECT START DATE
 I030     N06EST-END-DATE ESTIMATED COMPLETION DATE
 I100M000TTLLINES-IN      LINES OF SOURCE CODE INPUT
** MDFORMAT  LEVEL=SUBSYSTEM,OP=ADD
 I010S001TTLNBR-MODULES  NUMBER OF MODULES
 I020M070TTLNBR-UNITS    NUMBER OF UNITS
 I100     N04P-NBR-UNITS  PLANNED NUMBER OF UNITS
** MDFORMAT  LEVEL=MODULE
                  .
                  .
                  .
** MDUPDATE  UNIT=TIPTOP-SYSTEM,LEVEL=SYSTEM,OP=ADD      {WRITE MANUAL INPUT
 010=TEST PROJECT DESCRIPTION                           {DATA
 020=022878
 030=050178
** MDUPDATE  UNIT=TIPTOP-SUBSYSTEM,OP=ADD
 100=0010
 ** MDXCHECK UNIT=TIPTOP-SUBSYSTEM                       {CHECK FOR NUMBER OF
 ** INSERT                                               {UNITS CREATED
   020>100:V01,041578,MORE UNITS THAN PLANNED
 @END  PSL
 @XQT   PSL.BCTL
 @FIN
```

Input source data for unit TIPTOP and its included units. Correct the included unit TIPTOP-ENVIRONMENT-DIVISION. Move a copy of TIPTOP to the integration library. Save the source data on tape.

```
@RUN     ......
@ASG,A     DMA*PSL.
@USE       PSL.,DMA*PSL.
@ADD,L     PSL.RUN
** PARAM PROJ=5I24SSSPSL,LIBR=DEVEL
** ADD  SECT=SOURCE,UNIT=TIPTOP,LANG=SCOBOL,PGMR=SMITH      CREATE MAIN UNIT
       IDENTIFICATION DIVISION.
           PROGRAM-ID.  TIPTOP.

           INCLUDE TIPTOP-ENVIRONMENT-DIVISION.

       DATA DIVISION.

           INCLUDE TIPTOP-FILE-SECTION.

           INCLUDE TIPTOP-WORKING-STORAGE

       PROCEDURE DIVISION.

           INCLUDE TIPTOP-PROCEDURE-DIVISION.

** ADD     UNIT=TIPTOP-ENVIRONMENT-DIVISION                 CREATE INCLUDED UNITS
       ENVIRONMENT DIVISION.


         SOURCE-COMPUTER.
           UNIVAC-1108.
         OBCT COMPUTER.
           UNIVAC-1108.

        INPUT-OUTPUT SECTION.
              .
              .
              .
** ADD   UNIT=TIPTOP-FILE-SECTION.
```

```
.
.
.
** CHANGE UNIT=TIPTOP-ENVIRONMENT-DIVISION,PGMR=SMITH        CORRECTIONS TO UNIT
** MODIFY  LINE=6,FROM=OBCT,TO=OBJECT
** INSERT  AFTER=2
          CONFIGURATION SECTION.
** MOVE  UNIT=INTEG-TIPTOP,LIBRARY=INTEG                     {PLACE TOP UNIT IN
** SECT=SOURCE,OLDUNIT=TIPTOP,OLDL=DEVEL                     {INTEGRATION LIBRARY
** BACKUP  LIBR=DEVEL,SECT=SOURCE                           WRITE SOURCE SECTION TO TEMP DIS
@END  PSL
@XQT PSL.BCTL
```

```
@ASG,CP <file-name>,<storage-mode>
@COPY UT,<file-name>
@FIN
```

where:

<filename> is any unique UNIVAC filename
<storage-mode> may be either tape or disc

Restore source data from tape. Collect management data items. Print two different management reports:

1) A Program Structure of unit TIPTOP
2) The management data report previously defined and collected.

```
@RUN
@ASG,A DMA*PSL.
@USE PSL,DMA*PSL.
@ADD PSL,RUN
** PARAM PROJ=5124SSSPSL,LIBR=DEVEL
** RESTORE  SECT=SOURCE                          . READ SOURCE SECTION FROM TEMP DISC
** MDCOLLECT                                     . COLLECT MANAGEMENT DATA
** MDPRINT  SECT=SOURCE,UNIT=TIPTOP,REPORT=PS    . PRINT PROGRAM STRUCTURE OF UNIT TIPTOP.
** MDPRINT  REPORT=MD,UNIT=YES,SECT=MGMT         . PRINT MANAGEMENT DATA REPORT
@END   PSL                                       . ALLOCATE BACKUP FILE
@ASG,A <file name>                               . RESTORE BACKUP FILE TO DISK
@COPY <file name>,RT
```

where:

<file name> is the name of the backup file

```
Compile unit TIPTOP.  Create LINK unit LNKTIP and JOB unit XQTTIP, then map and
execute TIPTOP.
@RUN -----
@ASG,A DMA*PSL
@USE PSL, DMA*PSL
@ADD     PSL.RUN
**  COMPILE  PROJ=5124SSSPSL,LIBR=DEVEL,      TIPTOP COMPILATION
**                UNIT=TIPTOP
** ADD     SECTION=LINK,UNIT=LNKTIP           COLLECTOR CARDS FOR TIPTOP
       INCLUDE   TIPTOP                       ABSOLUTE ELEMENT NAME IS LNKTIP
       INCLUDE   CALLST                       CALLST IS A CALLED STUB UNIT

** ADD    SECTION=JOB,UNIT=XQTTIP             JCL FOR TIPTOP
@ASG,T    CARDIN,F///10
@ASG,T    PRINTOUT,F///20
@XQT                                                      EXECUTE
@DATA,L   PRINTOUT.
@END
** LINK    UNIT=LNKTIP                        MAP LNKTIP
** EXECUTE  JOB=XQTTIP,LOAD= LNKTIP          EXECUTE LNKTIP
@END  PSL
@ASG,T  CARDIN,F///10
@DATA,I CARDIN.                              {DATA CARD INSERTED
  TEST DATA CARD 1                           {INTO CARDIN
  TEST DATA CARD 2
          .
          .
          .
@END
@XQT    PSL.BCTL
@FIN
```

Print various reports scanning different sections of DEVEL or INTEG.

```
@RUN  -----
@ASG,A DMA*PSL.
@USE PSL.DMA*PSL.
** AUTHOR   PROJ=5124SSSPSL,LIBRARY=DEVEL,          {LIST ALL SOURCE UNITS
**       SECTION=SOURCE,OPTION=SOURCE,              {WRITTEN OR MODIFIED BY
**       OPGMR=SMITH,UPGMR=SMITH                    {PROGRAMMER SMITH
** CSCAN  SECTION=LOAD,STRING="CALLST"              LOCATE LOAD UNITS CONTAINING CALLST
** DOCUMENT  SECTION=SOURCE,LSPACE=2,PLINES=30      PRINT INCLUDED UNIT
** HEADER
       INCLUDED UNIT OF TIPTOP
**  TEXT     UNIT=TIPTOP-ENVIRONMENT-DIVISION
**  INDEX  SECTION=JOB                              LIST NAMES OF ALL UNITS
**  INDEX  LIBR=INTEG,SECT=SOURCE                   IN SECTION
**  SOURCE   UNIT=ALL                               LIST ALL SOURCE UNITS
@END PSL
@XQT   PSL.BCTL
@FIN
```

SECTION 4. NON-ROUTINE OPERATIONS

The entire PSL system in source, relocatable and absolute form is contained in a single program file named DMA*PSLPRG. This file contains the source code, acceptance test and installation test data, JCL used to maintain and execute the PSL system plus an independently executable COBOL Precompiler. The program file is normally kept only on tape and is restored only when modifications are to be made to the system. This file is normally referred to by the use relationship PSLPRG.

The installed version or 'floor' version of PSL uses the program file DMA*PSL. to contain the absolute form of the PSL system. In addition there are four related files which complete the floor version of PSL. These are:

DMA*XPROJECT.
DMA*JPSL.
DMA*SPSL.
DMA*PSL-STUB.

These files, used by the PSL-system to support spawned job operations were generated using the facilities of the BCTL module. The floor version of PSL is normally referred to by the USE relationship PSL.

This section details the following Non-routine operations:

a. System Backup
b. System Restore
c. System Maintenance

4.1 System Backup

The PSL system is backed up onto tape by using the Univac 1100 series File Utility Routine and Program File Utility Routine (FURPUR). @COPY,GM commands are used to copy the PSL files to tape (see Figure 1).

4.2 System Restore

Similarly, @COPY command can be used to restore the PSL system from tape (see Figure 2).

4.3 System Maintenance

If it becomes necessary to modify PSL code or add new modules, the program file DMA*PSLPRG must be restored from tape. The @COPY command can be used to create a working copy ot the PSL system under a new Project ID normally a working file called DMA*PSL.MODS is allocated to contain a version of the PSL system. This file is referred to by the USE relationship M. in the JCL contained in DMA*PSLPRG. Corrections are then made using the Univac 1100 or University of Maryland editor. The majority of the PSL modules are written in Structured COBOL, and these modules must be

```
@RUN              <RUN ID>,<ACCOUNT-#>,<PROJECT ID>
@ASG,CP           SAVE*PSL.,U9V,<reel-number>
@ASG,CP           DMA*PSLPRG.
@ASG,A            DMA*PSL.
@ASG,A            DMA*XPROJECT.
@ASG,A            DMA*JPSL.
@ASG,A            DAM*SPSL.
@ASG,A            DMA*PSL-STUB.
@COPY,GM          DMA*PSLPRG.,SAVE*PSL.
@COPY,GM          DMA*PSL.,SAVE*PSL.
@COPY,GM          DMA*PROJECT.,SAVE*PSL.
@COPY,GM          DMA*JPSL.,SAVE*PSL.
@COPY,GM          DMA*SPSL.,SAVE*PSL.
@COPY,GM          DMA*PSL-STUB.,SAVE*PSL.
@FIN
```

FIGURE 1. BACKUP PROCEDURE

```
@RUN      <run-id>,<account-#>,<project-id>
@ASG,A     SAVE*PSL.,U9V,<reel-number>
@ASG,A    DMA*PSLPRG.
@ASG,A    DMA*PSL.
@ASG,A    DMA*XPROJECT.
@ASG,A    DMA*JPSL.
@ASG,A    DMA*SPSL.
@ASG,A    DMA*PSL-STUB.
@COPY,G   SAVE*PSL.,DMA*PSLPRG.
@COPY,G   SAVE*PSL.,DMA*PSL.
@COPY,G   SAVE*PSL.,DMA*XPROJECT.
@COPY,G   SAVE*PSL.,DMA*JPSL,
@COPY,G   SAVE*PSL.,DMA*SPSL.
@COPY,G   SAVE*PSL.,DMA*PSL-STUB.
@FIN
```

FIGURE 2. RESTORE PROCEDURE

precompiled as shown in Figure 3. After the changed modules have been recompiled, the system must be collected. The collector control cards are stored in PSLPRG.BCTLMAP. If new modules have been added to the system, the elements must be listed in PSLPRG.BCTLMAP. The system can then be collected (see Figure 4). It is recommended that both the new PSL and the new PSLPRG be backed up onto tape.

```
@RUN    -----
@ASG,A      DMA*PSLPRG.
@USE        PSLPRG.,DMA*PSLPRG.
@ASG,T          PR,F///1
@ASG,T          PI,F///100
@ASG,T          PO,F///100
@ASG,T          LS,F///100

@DATA,I   PI.
@ADD,D      PSLPRG.<element name>
@END
@XQT        PSLPRG.PRECOMPILER
@ACOB,ITVSEC mTPP$,PSLPRG PSLPRG                    <element name>

@ADD        PO.
@EOF

@FIN
  where:
     T = Reverses display meaning defaul' fieldata
     V = Subproblem
     S = Source list of
     PC = Output all diagnostics
     C = List library text
     M = Suppress the COBOL monitor option
     Z = compile debug on
     <element name> = name of element to be changed
```

Temporary Files required by the COBOL Precompiler

Precompile Input File

Precompiler Output File

FIGURE 3. STRUCTURED COBOL COMPILATION

```
@RUN        -----
@ASG,A      DMA*PSLMODS.
@USE        M.,DMA*PSLMODS.

@MAP        PSLPRG.BCTLMAP,M .BCTL      } MAP options may be used
                                        } (Except I option)
@EOF
```

FIGURE 4. PSL SYSTEM COLLECTION