AUTOMATED DOCUMENTATION SYSTEM (ADS)
STUB GENERATOR: DESCRIPTION AND
USER INSTRUCTIONS

(Comprehensive Standard for Software Development)

by
Linda Lawrie
Jean Baugh

Approved for public release: distribution unlimited.
This report describes a subsystem of the Automated Documentation System (ADS) called the Stub Generator. The Stub Generator lets software development managers define which ADS documentation sections must be in the final software code. It also lets the manager monitor updated ADS documentation as initial software codes are amended and extended. The programmer is given a convenient way of developing a system in a top-down structural manner through the generation of standard program stubs.
Block 20 continued.

This report is written for the data processing professional and assumes the reader is familiar with Control Data Corporation (CDC) FORTRAN and UPDATE and the ADS computer system as described in the U.S. Army Construction Engineering Research Laboratory (CERL) Technical Report E-147, *The Automated Documentation System — User Manual* (CERL, February 1979).
FOREWORD

This research was conducted by the Energy Systems Division (ES) of the U.S. Army Construction Engineering Research Laboratory (CERL) for the Engineering and Scientific Division of the Engineering and Data Systems Office, Department of the Army, under RDT&E Program 6.27.25A, "Engineering Software Development," Task 02, "Comprehensive Standard for Software Development." Mr. Gene Manning was the Technical Monitor.

Mr. Douglas C. Hittle was the CERL Principal Investigator. Administrative support was provided by Dr. D. J. Leverenz and Mr. R. G. Donaghy, Chief of CERL-ES.

The ADS Stub Generator program was authored by Ms. J. Baugh and Ms. L. Lawrie. Appreciation is expressed to Ms. M. L. Scala for her help in writing this report.

COL Louis J. Circeo is Commander and Director of CERL, and Dr. L. R. Shaffer is Technical Director.
CONTENTS

DD FORM 1473 ........................................... 1
FOREWORD ........................................ 3

1 INTRODUCTION ........................................... 5
   Background
   Objective
   Mode of Technology Transfer

2 OVERVIEW ................................................ 5
   First Session
   History File
   Creating Program Modules
   File List and Function Description
   Options
   Help

3 CAPABILITIES AND LIMITATIONS ..................... 8
   Flexible Features
   Limitations
   Output

4 COMMAND LANGUAGE AND ONLINE HELP .............. 9

5 GENERAL HINTS ........................................ 10

6 SUMMARY ............................................... 11
   APPENDIX A: Sample First Session .................. 12
   APPENDIX B: Sample Run ............................ 17
   APPENDIX C: Instructions for Beginning a Stub  
               Generator Session on BCS ............... 25
   APPENDIX D: Batch Job Output ..................... 26

DISTRIBUTION
AUTOMATED DOCUMENTATION SYSTEM (ADS) STUB GENERATOR: DESCRIPTION AND USER INSTRUCTIONS

1 INTRODUCTION

Background
A useful software tool is the product of a carefully managed and executed software development project.

In 1979, the U.S. Army Construction Engineering Research Laboratory (CERL) introduced a computer system to help guarantee successful management and execution of Army software development. This system, called the Automated Documentation System (ADS), allows internal and external documentation to proceed simultaneously with software development.1

The ADS Stub Generator is a subsystem of ADS. It lets the software project manager define which ADS documentation sections must be in the final software code. It also lets the manager monitor updated ADS documentation as the initial software code is amended and extended.

This report describes how to use the ADS subsystem Stub Generator. The text is geared to the data processing professional and assumes the reader is familiar with Control Data Corporation (CDC) FORTRAN and UPDATE and the ADS computer system as described in CERL Technical Report (TR) E-147.

Objective
The objective of this study was to develop a method to ensure structured program development and adequate documentation by automatically generating standard program stubs.

Mode of Technology Transfer
The ADS Stub Generator is available from the Boeing Computer Services (BCS) Company under the U.S. Army Corps of Engineers contract for scientific and engineering teleprocessing.


2 OVERVIEW

Stubs are skeletal routines which contain routine headers, preliminary ADS documentation, type statements, common block references, initial debugging output, and RETURN/END statements. The Stub Generator relieves some of the drudgery of creating these routine stubs by building skeleton FORTRAN modules containing standard UPDATE directives, basic FORTRAN code, and ADS comments from data entered by the user. These skeleton modules are kept in the UPDATE source program library. They can be amended or extended as the system is developed.

First Session
During the first Stub Generator session, the user (usually the software project team leader) sets the desired level of internal program documentation by specifying the required ADS documentation sections (see Appendix A). At this time, the team leader can also set basic "system parameters" for the software being developed; once set, some of these system parameters cannot be changed. For example:

1. The name of the system (in the form of a three-character prefix which is used to generate file names)

2. The password (if any) associated with system files

3. The control character to be used on UPDATE directives.

A complete list of parameters which can and cannot be changed after the first Stub Generator session is in Appendix A, Figure A1. Appendix A also gives a sample of an initial Stub Generator session.

History File
The system history file is maintained by the Stub Generator. It has all the data needed to describe a software system to the Stub Generator (Figure 1). It includes:

1. A list of all options in effect during the last Stub Generator session

2. Data on modules already named or described to the Stub Generator subsystem

3. Data needed to create instructions to update and sort the source program library.
Creating Program Modules

During the creation of program modules, the Stub Generator prompts the programmer for information needed to define each program module. The Stub Generator ensures adequate internal program documentation by:

1. Prompting with the ADS section headers in the order specified by the software manager during the first session.

2. Verifying the specified ADS section for the type of module being defined. After the required ADS comments are given to the Stub Generator, the programmer may input additional ADS sections, if desired.

Two UPDATE files can be created during each Stub Generator session. One has preliminary UPDATE directives (such as *ID or *AF) and any modules defined as COMDECKS. The other has UPDATE directives and source code for any modules defined as DECKS.

Both files are kept on disk until a programmer tells the Stub Generator to put them in the source program library. (For more details, see the explanation of the @D command in Chapter 4.)

After the Stub Generator is told to update the source program library, it generates a file of control cards. These cards are used to begin the proper job stream. (This file is sent to the job queue internally. However, the Stub Generator will save this file as
a disk file to be batched by hand, if requested.) Once
the job begun by these control cards is complete, any
update files will be purged; the newly sorted source
program library will be saved and a printed report of
the final source library will be output.

If the option to run ADS is in effect when the
control card file is created, the control card job will
also create and keep two ADS files. It will also
produce any ADS reports requested.

A sample programmer session is given in Appendix
B. (Actual usage of the Stub Generator will differ
from facility to facility. Commands for beginning a
session on Boeing Computer Services' (BCS) EKS
system are given in Appendix C.)

**File List and Function Description**

Figure 1 shows the overall file system for the
Stub Generator.

The system help file contains the online help
messages that are given in response to the @H
command (Chapters 4 and 5).

The system error message file contains the online
help messages pertaining to error messages issued by
the Stub Generator. These messages are given in
response to the @M command (see Chapters 4 and 5).

The control card file contains the job stream
that is generated and submitted by the Stub Genera-
tor. This file may be saved as a permanent disk
file. If the user asks the Stub Generator to save the
control card file, the permanent file name will be
generated by concatenating the user-supplied prefix
with CNTL (for three-character prefixes), OCNTL
(for two-character prefixes), and LOCNTL (for one-
character prefixes). The batch job that is submitted
will generate printed output and a sorted UPDATE
source program library (OLDPL). The OLDPL con-
tains all of the source code for the software system in
a special UPDATE format.

The ADS master and invert files will also be
created if the RUN ADS option is in effect. These
files contain information about the code in the source
program library needed to generate ADS reports.

**Options**

The Stub Generator optionally inserts initial debug-
ging features when the skeleton modules (Figure 2)
are created. These features "trace" program paths and
parameter passing.

The initial debugging features are controlled by the
logical variables TRACE and PARAMS. These vari-
bles (along with the variable DEBUG*) are assumed
to be in the standard debug common block of the
software system being developed. The TRACE and
PARAMS flags can be turned on or off internally
during debugging. Besides TRACE, PARAMS, and
DEBUG, the Stub Generator assumes that the soft-
ware system has a routine called TRACER.

Using these variables and the routine TRACER,
the Stub Generator automatically creates the code to
trace program paths and display passed parameters at
routine entries and exits.

**Help**

Two system message files explain error messages
and tell the user what the causes of the error might be.
The online help features are invoked by the @H and
@M commands. (For details, see Chapters 4 and 5.)

```
SUBROUTINE SUBI(X,Y)

IF (TRACE) CALL TRACER (4HSUBI,1)
IF (PARAMS) WRITE (DEBUG,*),X,Y

IF (PARAMS) WRITE (DEBUG,*),X,Y
IF (TRACE) CALL TRACER (4HSUBI,2)
RETURN
END
```

*Figure 2. Sample Stub Generator debugging statements.*
3 CAPABILITIES AND LIMITATIONS

The Stub Generator automatically generates all the UPDATE directives needed to (1) create a source program library, (2) add new modules, or (3) purge existing modules. As module names are mentioned (either as the module is defined or in a context which identifies the module type), it builds a history of names to warn users of possible conflicts (e.g., using duplicate module names, using the same name in a context that implies different routine types, or creating a subroutine call to a defined subroutine whose argument list has a different number of arguments than the defined subroutine).

For example, if a new module is given the same name as a module already in the system, the Stub Generator will inform the user that the named module exists and ask if he/she wants to purge the existing version. The history may also detect usage conflicts, such as naming a module previously defined as a function in a generated subroutine call, and a few possible error conditions, such as a conflict between the number of formal parameters given in the module definition and the number of parameters given in the argument list of a generated subroutine call to that previously defined module.

Flexible Features

1. The Stub Generator accepts card image input for ADS comments. This lets the user exercise all the formatting capabilities in ADS.

2. Generated subroutine calls and user-defined code (e.g., comments, calculations, and function calls) can be put in any order in the body of the routine. The Stub Generator asks for subroutine calls and other code and requires consecutive negative responses to both requests before completing the module. The Stub Generator will automatically include "RETURN" or "STOP" and "END" statements as needed.

3. The UPDATE source library (OLDPL) is output in a standard order of COMDECKS, main programs, and subroutines and functions. To ask for a nonstandard ordering, the user assigns a level from 1 to 3 to each module when it is defined. The OLDPL will then be created in the order:

   a. COMDECKS
   b. Level 3 routines
   c. Level 2 routines
   d. Level 1 routines.

Each subgroup of either OLDPL ordering is kept in alphabetical order. Module levels are stored in the Stub Generator history file and their order maintained as modules are added or deleted.

4. Permanent files can be accessed under an alternate user number. Thus, the OLDPL and the basic system history file can be generated and maintained under a master account, and be accessed by several different subsidiary accounts.

5. A full update of the ADS master files is always done when an ADS run is requested. This ensures that all possible changes are included. (The Stub Generator creates an ADS batch job to do this; see pp 6, 7.) But if the user is working with a pre-existing OLDPL (or if there are a number of completed modules), it may be better to run ADS jobs more tailored to the specific update needs outside of the Stub Generator.

Limitations

1. The Stub Generator has no file editing capabilities other than a very limited line replacement for the job control stream and backspacing one input line. Any big change should be made by a separate UPDATE run or by redefining the defective module.

   A module is redefined by entering its name in response to the request for the next routine name and by allowing the STUB Generator to purge the old routine. (Modules cannot be purged or redefined on the first run, since the purge directive is not valid when a new OLDPL is being created.) The Stub Generator does not interactively update the OLDPL; rather, it generates input files of directives and creates a batch job from a user session to accomplish the actual OLDPL modification.

2. Because the Stub Generator is not a FORTRAN compiler, a clean compile is not guaranteed by generating a module. For example, if the software system being developed is to automatically include TRACER calls, the user must define "TRACE" as a logical variable in a standard debug COMDECK which must be included in each of the system's routines. If this is not done, a FORTRAN compile error will occur. To include automatic unformatted parameter writes, "PARAMS" must be defined in the same way.

*The output unit for the software system's debugging output.
as TRACE, TRACE, PARAMS, and the standard debug COMDECK refer to the special initial debugging outlined in the overview. The debug COMDECK name is defined during the first Stub Generator session and maintained in the Stub Generator history file (see Chapter 2).

Figure 2 is a sample of the debugging statements created by the Stub Generator.

**Output**

Batch job output lists (1) all active cards in the OLDPL, (2) any FORTRAN compilation errors, and (3) any ADS reports requested. If an ADS run was not requested, only the OLDPL list will be output. A sample batch job submitted by the Stub Generator is given in Appendix D.

### 4 COMMAND LANGUAGE AND ONLINE HELP

Nine commands ask for special action from the Stub Generator. The command syntax is a "@" immediately followed by a one-letter command (or the full command word). Table 1 explains each command.

The online help commands (@H and @M) tell the user what type of information the Stub Generator is asking for or what type of errors might account for a particular error message.

The @H command gives clues about the type of information the Stub Generator is currently request-

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Command</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>@A</td>
<td>@ABORT</td>
<td>Aborts the Stub Generator program without saving any of the current work files.</td>
</tr>
<tr>
<td>@B</td>
<td>@BACKSPACE</td>
<td>Backspaces one record on the last file written. This command must be used carefully. It may not be possible to backspace to the correct place since the Stub Generator maintains several files simultaneously. If a different prompt has intervened since the record to be replaced was displayed, (e.g., I, M, !), the backspace command may not work properly. However, errors can be fixed by doing a standard UPDATE run.</td>
</tr>
<tr>
<td>@C</td>
<td>@COMMANDS</td>
<td>Lists the available commands; this list is displayed on the terminal.</td>
</tr>
<tr>
<td>@D</td>
<td>@DONE</td>
<td>Begins normal end-of-job processing to generate and submit a batch job to update the OLDPL (and run ADS, if requested).</td>
</tr>
<tr>
<td>@E</td>
<td>@ECHO</td>
<td>Toggles echoing of lines input.</td>
</tr>
<tr>
<td>@H</td>
<td>@HELP</td>
<td>Asks for online help giving more information on the question currently being asked of the user.</td>
</tr>
<tr>
<td>@M</td>
<td>@MESSAGE</td>
<td>Asks for online help giving more information on the last error message given to the user.</td>
</tr>
<tr>
<td>@M,####</td>
<td>@MESSAGE,####</td>
<td>Asks for online help for a particular error message identified by a four-digit code (####).</td>
</tr>
<tr>
<td>@P</td>
<td>@PRINT</td>
<td>Prints the last line input by the user.</td>
</tr>
<tr>
<td>@S</td>
<td>@SAVE</td>
<td>Saves the user's work files and run options without beginning a batch run. The user can then input more definitions later.</td>
</tr>
</tbody>
</table>
ing. It also tells the user what publications to refer to for more detailed explanations or syntactical rules.

The @M command has two forms: both give extended error messages. (Extended error messages give a detailed statement of the error, possible causes, and names of appropriate reference manuals.) Entering "@M" will result in an extended error message for the last displayed error message. Extended error messages may be retrieved by typing "@M,dddd," where "dddd" is the four-digit code preceding the brief message for which the user is requesting additional information.

Up to three levels of help may be available for any particular question or error message. A "+" at the end of a help section tells the user that additional help is available.

5 GENERAL HINTS

1. A null line (RETURN) is the same as a negative response for Y/N questions, except during the first run of the Stub Generator, when basic system parameters are defined. During this first run, the default for all parameters is Y.

2. The Stub Generator will output an UPDATE input file for an existing OLDPL when the user exits a session with the @S command. The user should then edit the UPDATE input files to make sure the *AF directives indicate the correct order for the OLDPL. The permanent file names of the update input files are created by concatenating the user-supplied prefix with DTEI and DTES (for a three-character prefix), PDTEI and PDTES (for a two-character prefix), or UPDTEI and UPDTES (for a one-character prefix). For example, if the software prefix is "SAM," the UPDATE input files are SAMDTEI and SAMDTES. The user can then create a new job stream to begin the UPDATE run.

3. Abbreviations can be used to name both required and optional ADS sections. Valid abbreviations are usually made from the first three characters of the section name (Table 2).

4. When the user ends a list format with a comma, it tells the Stub Generator that the user wants to add more data to the list. For example, if a list of COMDECKS called in a routine is entered as "COM1, COM2, COM3," the Stub Generator will ask the user for another input line. The user then can add more COMDECK names.

5. If the batch job submitted by the Stub Generator fails for any reason, the user should try to resubmit a corrected run before initiating a new Stub Generator session. The exception is when an abort has occurred after an OLDPL manipulation. In this case, the user should try to resubmit a corrected run before initiating a new Stub Generator session.

<table>
<thead>
<tr>
<th>Category</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>TIT</td>
</tr>
<tr>
<td>Common block title</td>
<td>COM TIT</td>
</tr>
<tr>
<td>Author</td>
<td>AUT</td>
</tr>
<tr>
<td>Date Written</td>
<td>DAT</td>
</tr>
<tr>
<td>References</td>
<td>REF</td>
</tr>
<tr>
<td>Location</td>
<td>LOC</td>
</tr>
<tr>
<td>Method</td>
<td>MET</td>
</tr>
<tr>
<td>Control cards</td>
<td>CON</td>
</tr>
<tr>
<td>Remarks</td>
<td>REM</td>
</tr>
<tr>
<td>System</td>
<td>SYS</td>
</tr>
<tr>
<td>Flow</td>
<td>FLO</td>
</tr>
<tr>
<td>Files</td>
<td>FIL</td>
</tr>
<tr>
<td>Algorithm</td>
<td>ALG</td>
</tr>
<tr>
<td>System dependencies</td>
<td>SYS DEP</td>
</tr>
<tr>
<td>Nonsystem externals</td>
<td>NON SYS EXT</td>
</tr>
<tr>
<td>Machine dependencies</td>
<td>MAC DEP</td>
</tr>
<tr>
<td>Implementation dependencies</td>
<td>IMP DEP</td>
</tr>
<tr>
<td>Variable dictionary</td>
<td>VAR</td>
</tr>
<tr>
<td>Revised (date)</td>
<td>REV</td>
</tr>
<tr>
<td>Purpose</td>
<td>PUR</td>
</tr>
</tbody>
</table>
case, the user should purge the work files. Any missing output can then be retrieved by beginning the appropriate job outside of the Stub Generator. If the output is not retrieved in this way, it will be produced during the next successful Stub Generator run.

6. Three types of prompts are produced by the Stub Generator and give a clue to the type of input expected:

D> This is the standard prompt indicating that a reply to the current question is desired. The type of input desired is generally indicated in the question itself or in the online help (@H).

M> This prompt indicates the Stub Generator is ready to accept multiple input lines. The lines will be accepted as 80-column card image lines. No processing of the line will be done other than to write it to the appropriate file.

CD This prompt indicates the Stub Generator is ready to accept a card image input of an ADS comment. The CD is used as columns 1 and 2 of the line written and need not be repeated by the user.

7. The Stub Generator gives two classes of error messages: warning and error. Warning messages generally tell the user the Stub Generator has detected a special condition and has taken corrective action, is expecting special input (i.e., continuing input following a line ending with a comma), or has detected a condition that may result in an error (i.e., an incorrect number of arguments on a subroutine call to an already defined subroutine). Fatal error messages indicate a severe error that must be corrected and re-input before processing can continue.

6 SUMMARY

The Stub Generator gives the software development manager who relies on ADS a way of standardizing documentation during all phases of project development. By setting specific, unchangeable ADS documentation standards during the first Stub Generator session, the manager guarantees that minimum documentation goals will be met, no matter how often the software is amended or extended. The Stub Generator also lets the manager monitor project progress easily and reliably. The programmer is also given a convenient way of creating program systems in a top-down structural manner through the generation of standard program stubs.
APPENDIX A:
SAMPLE FIRST SESSION

The parameters which define the basic characteristics of the software being developed are set during the first Stub Generator session. This first session is usually done by the software development team leader. The team leader can control the system characteristics and the basic ADS documentation by setting parameters which cannot be changed. In the sample session in this appendix, the system parameter library was saved by using the @S command. The team leader could also choose to define the standard debug common block and the TRACER routine to be used by the system and begin the batch run to create an OLDPL by ending the session with the @D command.

The prompt to the user is “I>”. If nothing is shown on that line, the user used only the RETURN key. This key defaults to “YES” or the value indicated in Figure A1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Change</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master Account</td>
<td>None</td>
<td>X</td>
<td>User id under which system parameters were defined and where OLDPL and system library are maintained by system.</td>
</tr>
<tr>
<td>System Prefix</td>
<td>None</td>
<td>X</td>
<td>3 user-specified characters that are appended to the front of standard file names to create names uniquely identifying files belonging to a particular system.</td>
</tr>
<tr>
<td>Password</td>
<td>Null</td>
<td>X</td>
<td>Password associated with system-maintained permanent files.</td>
</tr>
<tr>
<td>Update Control Character</td>
<td>*</td>
<td>X</td>
<td>Character appended to the front of UPDATE directives.</td>
</tr>
<tr>
<td>ADS Required Sections</td>
<td>None</td>
<td>X</td>
<td>User-specified list of ADS sections that are to be generated for each module defined. Specified headers will always be generated even if no input is received for that section.</td>
</tr>
<tr>
<td>Account Password</td>
<td>Null</td>
<td>X</td>
<td>Password associated with user id under which user is currently running.</td>
</tr>
<tr>
<td>List Control</td>
<td>Yes</td>
<td>X</td>
<td>Controls generation of **‘IF -DEF, (mod name),1’, ‘C: LIST.NONE’ and ‘IF DEF.LISTAIL’, ‘C: LIST.AIL’, statements to allow control of source listing in compiler output.</td>
</tr>
<tr>
<td>Implicit Integer</td>
<td>Yes</td>
<td>X</td>
<td>Controls generation of &quot;IMPLICIT INTEGER (A-Z)&quot; statement.</td>
</tr>
<tr>
<td>Standard Debug Deck</td>
<td>Yes</td>
<td>X</td>
<td>User may request the generation of &quot;CALL&quot; to a user-specified debug comdeck in all DECKS defined.</td>
</tr>
<tr>
<td>Debug Deck Name</td>
<td>DEBUGR</td>
<td>X</td>
<td>Name of user-defined debug comdeck to be specified on &quot;CALL&quot;.</td>
</tr>
<tr>
<td>Standard OLDPL Order</td>
<td>Yes</td>
<td>X</td>
<td>Standard order = comdecks, main routines, subroutines/functions in alphabetical order within each group. User may request control of ordering of all modules except comdecks.</td>
</tr>
<tr>
<td>Nonpropagating</td>
<td>Yes</td>
<td>X</td>
<td>Controls generation of &quot;NOPROP&quot; parameter on &quot;COMDECK&quot; directive.</td>
</tr>
</tbody>
</table>

† Parameters that cannot be changed after the first session are "system parameters." Changeable parameters are "run-time options."

Figure A1. Stub Generator – first session sample.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Change</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRACER Calls</td>
<td>Yes</td>
<td>X</td>
<td>Controls generation of calls to a user-defined TRACER routine upon entry and exit to subroutines/functions. Provides a subroutine trace.</td>
</tr>
<tr>
<td>Unformatted</td>
<td>Yes</td>
<td>X</td>
<td>Controls generation of unformatted write statement with I/O list of formal parameters upon entry and exit to subroutines.</td>
</tr>
<tr>
<td>Parameter Trace</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comdeck Define</td>
<td>COMM</td>
<td>X</td>
<td>Name to be used in generation of &quot;IF DEF,&quot; directive preceding a block of ADS code</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td></td>
<td>in comdecks. Allows suppression of listing of comments.</td>
</tr>
<tr>
<td>Routine Define</td>
<td>ROUT</td>
<td>X</td>
<td>Name to be used in generation of &quot;IF DEF,&quot; directive preceding a block of ADS code</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td></td>
<td>in decks. Allows suppression of listing of comments.</td>
</tr>
<tr>
<td>ADS Run Initiation</td>
<td>Yes</td>
<td>X</td>
<td>Controls initiation of execution of the ADS program to generate update master documentation files and/or produce ADS reports. ADS runs will always redocument all modules defined in the old OLDPL. If the generator is being used to generate stubs for an existing system or only reports are desired, it would be more efficient for the user to initiate an ADS run separately.</td>
</tr>
</tbody>
</table>

† Parameters that cannot be changed after the first session are “system parameters.” Changeable parameters are “run-time options.”

**Figure A1.** (Cont’d.)
welcome to the bcs network
your access port is cix 67

select desired service: eks1

08/07/09. 07.51.30.

EKS1 7600.M0460.63AA 80/06/29.DS-0  18.35.11. 80/07/08.

-------------------------------

TERMINAL  27,  TTY
RECOVER/USER ID:  jeanb

*** NEW SCHEDULE FOR NOS 1.4 IMPLEMENTATION.  SEE MSG,NOS ***
C>get,profii/un=cer005
C>call,,jlist(master=cer004,sys=san,pw=test)
07.55.04. WELCOME TO JEAN’S INTERACTIVE STUDMAKER!
07.55.04. HANG ON WHILE I CRANK IT UP.

ADS STUD GENERATOR

INSTRUCTIONS?  (Y/N)

Y

THE ADS STUD GENERATOR SYSTEM IS DESIGNED AS AN AID TO A SYSTEMS
DESIGN TEAM IN THE INITIAL STAGES OF DEVELOPING A FORTRAN SYSTEM.
SYSTEM PARAMETERS ARE REQUESTED FROM THE USER TO DETERMINE OPTIONS
DESIRED, AND FOR USE WITH PERMANENT FILE FUNCTIONS. THESE OPTIONS ARE
RETAINED BETWEEN RUNS. SOME OPTIONS MAY BE CHANGED BY THE USER AT THE
BEGINNING OF A RUN.

MOST REQUESTS ARE SELF-EXPLANATORY. IF ADDITIONAL HELP IS REQUIRED,
THE USER MAY REFER TO THE USER’S GUIDE FOR MORE DETAILED INFORMATION,
OR REQUEST ON-LINE HELP FOR THE SPECIFIC QUESTION OR ERROR MESSAGE.
UP TO THREE LEVELS OF ON-LINE HELP ARE AVAILABLE. A "*" FOLLOWING A
SECTION OF TEXT INDICATES MORE HELP FOR THAT ITEM IS AVAILABLE AND
MAY BE RETRIEVED BY REPEATING THE SAME COMMAND (I.E., @H FOR
ADDITIONAL HELP ON THE CURRENT QUESTION, OR @M FOR ADDITIONAL
HELP ON AN ERROR MESSAGE).

AVAILABLE COMMANDS:
@A  @ABORT  ABORT PROCESSING
@B  @BACKSPACE  ONE RECORD ON LAST FILE WRITTEN
@C  @COMMANDS  PRINT LIST OF AVAILABLE COMMANDS
@D  @DONE  NORMAL END OF JOB PROCESSING
@E  @ECHO  TOGGLES ECHOING OF INPUT LINES
@H  @HELP  REQUEST ON-LINE HELP FOR CURRENT QUESTION
@M  @MESSAGE  REQUEST ON-LINE HELP FOR LAST ERROR MESSAGE ISSUED
@N,<<<  @MESSAGE,<<<  REQUEST OF ON-LINE HELP FOR SPECIFIED ERROR
          MESSAGE
@P  @PRINT  PRINT LAST LINE INPUT
@S  @SAVE  SAVE CURRENT FILES AND EXIT PROGRAM

**WARNING: 4700 INVALID TID
ABORT SESSION? (Y/N)
I> n
5 UPDATE CONTROL CHAR?
I> 6 ADS REQ SECTIONS? (NAME1,NAME2,...,NAME/NONE)
I> 7 ACCT PU?
I> 8 AUTO LIST CONTROL STMT? (Y/N)
I> 9 AUTO IMPLICIT INTEGER? (Y/N)
I> 10 AUTO STD DEBUG? (Y/N)
I> 11 DEBUG DECKNAME?
I> 12 STD OLDPL ORDER? (Y/N)
I> 13 AUTO NOPROP DECKS? (Y/N)
I> 14 AUTO TRACER CALLS? (Y/N)
I> 15 AUTO PARAMETER TRACE? (Y/N)
I> 16 ADS DECK DEF NAME?
I> 17 ADS ROUTINE DEF NAME?
I> 18 RUN ADS? (Y/N)
I> LIST SYSTEM PARANS? (Y/N)
I>y

SYSTEM PARAMETERS:

2 MASTER ACCT ID? : CERO84
3 SYS? (3 CHAR) NAME : SAN
4 PU? : TEST
5 UPDATE CONTROL CHAR? : *
6 ADS REQ SECTIONS?
16 ADS DECK DEF NAME? : COMM
17 ADS ROUTINE DEF NAME? : ROUT
LIST RUN OPTIONS? (Y/N)
I>y

RUN OPTIONS:

8 AUTO LIST CONTROL STMT? (Y/N) : Y
9 AUTO IMPLICIT INTEGER? (Y/N) : Y
10 AUTO STD DEBUG? (Y/N) : Y
11 DEBUG DECKNAME? : DEBUG
12 STD OLDPL ORDER? (Y/N) : Y
13 AUTO NOPROP CONDECKS? (Y/N) : Y
14 AUTO TRACER CALLS? (Y/N) : Y
15 AUTO PARAMETER TRACE? (Y/N) : Y
18 RUN ADS? (Y/N) : Y
ANY CHANGES? (Y/N)
I>y
ENTER ALL, # OR <CR>
I>6
6 ADS REQ SECTIONS? (NAME1,NAME2,...,NAME/NONE)
I>tit,com tit,aut,dat
ENTER ALL, # OR <CR>
I>
LIST SYSTEM PARAMS? (Y/N)
I>y

SYSTEM PARAMETERS:

2 MASTER ACCT ID?: CERO84
3 SYS? (3 CHAR MAX): SAM
4 PW?: TEST
5 UPDATE CONTROL CHAR?: 
6 ADS REQ SECTIONS?
   TITLE=
   COMMON BLOCK TITLE=
   AUTHOR=
   DATE WRITTEN=
16 ADS CONDECK DEF NAME?: COMM
17 ADS ROUTINE DEF NAME?: ROUT
LIST RUN OPTIONS? (Y/N)
I>n
ANY CHANGES? (Y/N)
I>n
MODULE NAME?
I>@s

OK. JUST A SECOND. I'M A SLOW WRITER.
08.00.42. COME AGAIN!
EXIT.
C>
APPENDIX B: SAMPLE RUN

The sample run in this appendix has examples of basic features like help commands and how to input ADS comments. A sample version of a TRACER routine and a debugging common block have been entered in this session and a batch run was begun to create an OLDPL. The job stream generated for the batch job is shown on p 24. This version uses KCL procedures, which restrict the range of valid UPDATE characters that can be used as the UPDATE control character. Only the characters A-Z, 0-9, and * can be passed as parameters to the procedures in the standard profile. The Stub Generator will only generate input files for UPDATE runs using other valid UPDATE control characters if the user creates a new job stream outside the Stub Generator.

Prompts to the user are "I>". "M>". and "CD". If nothing is shown on the prompt line. the user has entered only the RETURN.


ADS STUB GENERATOR

INSTRUCTIONS? (Y/N)
I>n
7 ACCT PW?
I>y
LIST SYSTEM PARAMS? (Y/N)
I>y

SYSTEM PARAMETERS:

2 MASTER ACCT ID?: CER084
3 SYS? (3 CHAR MAX): SAM
4 PW?: TEST
5 UPDATE CONTROL CHAR?: *
6 ADS REQ SECTIONS?
   TITLEZ=
   COMMON BLOCK TITLEZ=
   AUTHORZ=
   DATE WRITTENZ=
16 ADS CONDECK DEF NAME?: COMMON
17 ADS ROUTINE DEF NAME?: ROUT

LIST RUN OPTIONS? (Y/N)
I>y

RUN OPTIONS:

8 AUTO LIST CONTROL STMT? (Y/N): Y
9 AUTO IMPLICIT INTEGER? (Y/N): Y
10 AUTO STD DEBUG? (Y/N): Y
11 DEBUG DECKNAME?: DEBUGR
12 STD OLDPL ORDER? (Y/N): Y
13 AUTO NOPROP CONDECKS? (Y/N): Y
14 AUTO TRACER CALLS? (Y/N): Y
15 AUTO PARAMETER TRACE? (Y/N): Y
18 RUN ADS? (Y/N): Y

ANY CHANGES? (Y/N)
I>y
ENTER ALL, # OR <CR>
I>6

***ERROR: 2200 OUT OF BOUNDS
ENTER ALL, # OR <CR>
I>0m
NUMERIC VALUE OF USER INPUT EXCEEDS MINIMUM OR MAXIMUM ALLOWABLE VALUE.*
I>0m
EXAMPLE: USER INPUT=9 FOR OLDPL LEVEL, VALID VALUES ARE 1-3.
OR USER SPECIFIED A QUERY NUMBER TO CHANGE AN OPTION THAT
IS EITHER NOT 1-18 OR IS NOT CHANGEABLE ON THIS RUN
(SYSTEM PARAMETER).
LIST SYSTEM PARAMS? (Y/N)
LIST RUN OPTIONS? (Y/N)
ANY CHANGES? (Y/N)

MODULE NAME?
debugr

MODULE TYPE? (C,P,F,S)
C - CONDECK
P - PROGRAM
F - FUNCTION
S - SUBROUTINE

COMMON DEFINITION? (Y/N)

TYPE STATEMENTS? (Y/N)

DIMENSION STATEMENTS? (Y/N)

CD COMMON BLOCK TITLE=
debugr - debugging variable common

OPTIONAL SECTIONS? (SEC1,SEC2,...SECN/NONE)

VAR

OPTIONAL SECTIONS? (SEC1,SEC2,...SECN/NONE)

COMMON CALLS? (NAME1,NAME2,...,NAMEN/NONE)

OTHER CODE? (Y/N)

MODULE NAME?
tracer

MODULE TYPE? (C,P,F,S)
PARAMETER LIST? (P1,P2,...PN/NONE)

INPUT NAME, FLAG

TYPE STATEMENTS? (Y/N)

DIMENSION STATEMENTS? (Y/N)

ADS - REQUIRED SECTIONS

OPTIONAL SECTIONS? (SEC1,SEC2,...SECN/NONE)

PURPOSE?

VARIABLE DICTIONARY?

OPTIONAL SECTIONS? (SEC1,SEC2,...SECN/NONE)

COMDECK CALLS? (NAME1,NAME2,...,NAMEN/NONE)

SUBROUTINE CALLS? (Y/N)

OTHER CODE? (Y/N)

 other is calculated flag

if(flag .lt. 1.or. flag .gt. 2) goto 999

goto(901,902), flag
N> 901 continue
N>  enter subroutine
N>  tsptr = tsptr + 1
N>  write(debug,702) tsptr,?entered,sname
N>  goto 1000
N> 902 continue
N>  exit subroutine
N>  if (tsptr .le. 0) tsptr = 1
N>  write(debug,902) tsptr,?hexited,sname
N> 702 format(1h ,=(1h-),a7,1x,a10)
N>  tsptr = tsptr - 1
N> 1000 continue
N>  SUBROUTINE CALLS? (Y/N)
N>  OTHER CODE? (Y/N)
N>  MODULE NAME?
N>  MODULE TYPE? (C,P,F,S)
N>  FILE DEFINITIONS? (F1,F2,...FN/NONE)
N>  input, output, debug=output
N>  TYPE STATEMENTS? (Y/N)
N>  DIMENSION STATEMENTS? (Y/N)
N>  SUBROUTINE CALLS? (Y/N)
N>  PARAMETERS? (P1,P2,...Pn/NONE)
N>  param1,param2,param3,param4,param5,param6,param7,param8,param9,param10
N>  SUBROUTINE CALLS? (Y/N)

CD  TITLE=
CD  main - a sample main program
CD
CD  AUTHOR=
CD  y.a. programmer
CD
CD  DATE WRITTEN=
CD  1200 a.d.
CD
CD  OPTIONAL SECTIONS? (SEC1,SEC2,...SECN/NONE)
CD  none
CD
CD  CONDEC CALLS? (NAME1,NAME2,...,NAMEN/NONE)
CD  filecom
CD
CD  SUBROUTINE CALLS? (Y/N)
CD  y
CD
CD  NAME?
CD  sub1
CD
CD  PARAMETERS? (P1,P2,...PN/NONE)
CD  param1,param2,param3,param4,param5,param6,param7,param8,param9,param10
CD  SUBROUTINE CALLS? (Y/N)
I>y
M>c comments and other code can be interspersed with generated
M>c subroutine calls
M>
SUBROUTINE CALLS? (Y/N)
I>y
NAME?
I>sub2
PARAMETERS? (P1,P2,...PN/NONE)
I>none
SUBROUTINE CALLS? (Y/N)
I>n
OTHER CODE? (Y/N)
I>n
MODULE NAME?
I>sub2
MODULE TYPE? (C,P,F,S)
I>s
PARAMETER LIST? (P1,P2,...PN/NONE)
I>none
TYPE STATEMENTS? (Y/N)
I>n
DIMENSION STATEMENTS? (Y/N)
I>n
ADS - REQUIRED SECTIONS

CD TITLEx=
CD sub2 - a sample subroutine
CD
CD AUTHORx=
CD y.a.programmer
CD
CD DATE WRITTENx=
CD any date
CD
OPTIONAL SECTIONS? (SECI,SEC2,...SECN/NONE)
I>none
CONDECK CALLS? (NAME1,NAME2,...,NAMEm/NONE)
I>none
SUBROUTINE CALLS? (Y/N)
I>n
OTHER CODE? (Y/N)
I>n
MODULE NAME?
I>cd
**FILECOM**
THE ABOVE COMDECKS MUST BE DEFINED PRIOR TO INITIATING A RUN.
ENTER #8 TO END SESSION AND SAVE FILES, OR CONTINUE DEFINITIONS.
I>fileco
MODULE TYPE? (C,P,F,S)

22
**WARNING: 4400 TERMINATING ; SUPPLIED
**WARMING:
TITLE A SAMPLE ADS RUN GENERATED BY THE STUB GENERATOR

M>print wide dump for all commons;
M>print wide dump for all routines;
M>draw wide tree;
M>end;

1 /JOB
2 STUB6EN,T30,CM130000,P02.
3 USER,CERQIS,BLASTU. JEAND.
4 GET,PROFIL/UN=CERQIS.
5 GET,OLPL=SANQPL/UN=O,PV=TEST,NA.
6 CALL,UPDTE(OLP=SANQPL,UN=O,PV=TEST,CT=P,UC==)
7 CALL,OSRT(OLP=SANQPL,UN=O,PV=TEST,CT=P,UC==)
8 CALL,STADS(OLP=SANQPL,UN=O,PV=TEST,CT=P,UC==,INVT=SANINV,MAST=SANMAS)
9 PURGE,SANDTES.
10 PURGE,SANST.1.
11 PURGE,SANSCND/NA.
12 PURGE,SANSCCHND/NA.
13 EXIT, U.
14 COPYDF,UPLIST,OUTPUT.
15 IF(FILE(AWSOUT,LO))COPYDF,AWSOUT,OUTPUT,3.
16 EXIT.
17 /EDR
15

ANY CHANGES? (Y/N)

Do you wish to run this job? (Y/N)

Do you wish to save this deck? (Y/N)

08.31.35. Come again!

EXIT.

C> get, samcnt1

C> list, f=samcnt1

/JOB
STUBGEN,T30,CH130000,P02.
USER,CERO04,BLASTU.   JEAND.
GET,PROFIL/UN=CERO05.
GET,OLDPL=SANMPL/UN=0,PW=TEST,NA.
CALL,UPDTE(OPL=SANMPL,UN=0,PW=TEST,CT=S,UC=5)
CALL,OSRT(OPL=SANMPL,UN=0,PW=TEST,CT=S,UC=5)
CALL,STABS(OPL=SANMPL,UN=0,PW=TEST,CT=S,UC=5,INV=SANINV,MST=SANMST)
PURGE,SAMDTE.
PURGE,SAMDTE1.
PURGE,SANSCMD/NA.
PURGE,SAMTCMD/NA.
EXIT,U.
COPYBF,UPLIST,OUTPUT.
IF(FILE(ADSOUT,LO))COPYDF,ADSOUT,OUTPUT,3.
EXIT.
/EOR
/PACK
/READ,SANMTE.
/READ,SANMTE1
/EOR
*ID SORT
/EOR
YANK***
/READ,SAMTCMD
/EOR
YANK***
/EOF
*ID ADS
*DF CONN
*DF ROUT
*IF MAIN
*IF SUB2
*IF TRACER
/EOR
/READ,SANSCMD
/EOF
EOI ENCOUNTERED.
C>submit,samcnt1,ei=cero01
07.36.15. 00/07/14.LCIIBXZ
C>bye
JOB PROCESSING CCS  110.71
BYE 00/07/14. 07.36.30.
APPENDIX C:
INSTRUCTIONS FOR BEGINNING A STUB GENERATOR SESSION ON BCS

To begin a Stub Generator session on the Cyber 175 at BCS, the user should login and retrieve the procedure file:

GET.PROFIL/UN=CER005

The user then calls the interactive procedure:

CALL,,JIST(MASTER=<master account>,SYS=<system prefix>,PW=<password>,TID=<RJE user id>)

The parameter list for the call is:

<master account>: The ID under which the first session is/was executed to define the system parameters. Required so that the Stub Generator can access and maintain required permanent files.

<system prefix>: User-defined system identifier (maximum three characters).

<password>: Password to be associated with system-maintained permanent files. Needed to allow the Stub Generator to access and maintain the proper set of system files.

<RJE user id>: User ID to which output from the batch job is to be sent. This does not have to be specified if the user will not be beginning a batch job at the end of the session (@D command) — the resulting warning can be ignored. If the user fails to enter a TID but does wish to begin a run, the job should be aborted when requested and the procedure restarted with the proper parameters. The Stub Generator submits the batch job from the procedure; thus, an incorrect TID cannot be fixed from within the program.
This appendix lists output generated by the batch job begun by the Stub Generator session shown in Appendix B. The OLDPL listing shows the contents of the final sorted OLDPL. Immediately after the OLDPL listing is a short list of compile errors. These errors would not be listed if the batch job had not included an ADS run. The ADS output includes an echo of the commands received, a list of requested reports, and the requested tree of the system as it currently exists.
GENERATING UPDATE DIRECTIVES

DIRECTIVES FOR FILE "MAIN"

DIRECTIVES FOR FILE "FILECOM"

DIRECTIVES FOR FILE "TRACER"

UNLABELED OLDML CONNECTION IDENTIFIERS

update 1.3-498, 08/07/14, 07.36.38.

connection values are listed in chronological order of insertion

YANKEE DEBUG FILECOM TRACER MAIN SUB2

decks are listed in the order of their occurrence on a new program library if one is created by this update

DECKS ENCOUNTERED

YANKEE DEBUG FILECOM

DECKS WRITTEN TO COMPILE FILE

MAIN SUB2 TRACER

this update required 347008 words of core.
LIST OF CONTROL, ACTIVE, AND/or INACTIVE CARDS IN DEHUGR

DEHUGR *CDECK DEHUGR UMPI
DEHUGR END /DEHUGR DEHUGR, TRACE, PARMS
DEHUGR INTERN: DEHUGR
DEHUGR LOGICAL TRACE PARMS
DEHUGR C
DEHUGR #IF DEF,DEHUGR
DEHUGR CD COMM: FILE COMM: LOGIC TITLE=
DEHUGR CD COMM: FILE COMM: DEBUG IS VARIABLE COMMON
DEHUGR CD COMM: FILE COMM: VARIABLE DICTIONARY
DEHUGR CD COMM: FILE COMM: DEBUG FILE NAME
DEHUGR CD COMM: FILE COMM: TRACER LOGICAL VARIABLE FOR ROUTINE TRACING CALLS TO TIMER
DEHUGR CD COMM: FILE COMM: LOGIC VARIABLE FOR PARAMETER TRACING
DEHUGR #ENDIF

LIST OF CONTROL, ACTIVE, AND/or INACTIVE CARDS IN FILECOM

FILECOM *CDECK FILECOM UMPI
FILECOM COMM: FILECOM INPUT, OUTPUT
FILECOM #IF DEF,FILECOM
FILECOM CD COMM: FILE COMM: LOGIC TITLE=
FILECOM CD COMM: FILE COMM: FILE DEFINITION COMMON
FILECOM #ENDIF

LIST OF CONTROL, ACTIVE, AND/or INACTIVE CARDS IN MAIN

MAIN *CDECK MAINT
MAIN #IFDEF,MAINT
MAIN C/ LIST,NONE
MAIN #IFDEF,LIST, ALL
MAIN C/ LIST, ALL
MAIN PRAGMAS: OPTIMIZE, INPUT, OUTPUT, DEBUINPUT
MAIN IMPLICIT INTEGER (A-Z)
MAIN #IFDEF,OPTIMIZE
MAIN C/ TITLE=
MAIN CD MAIN: A SAMPLE MAIN, PRAGMAS
MAIN CD MAIN: MAIN
MAIN CD MAIN: MAIN
MAIN CD MAIN: MAIN
MAIN CD MAIN: MAIN
MAIN C/ TRACK GENERATOR
MAIN CD MAIN: MAIN
MAIN CD MAIN: MAIN
MAIN CD MAIN: MAIN
MAIN CD MAIN: MAIN
MAIN CD MAIN: MAIN
MAIN CD MAIN: MAIN
MAIN C/ END, MAINT

LIST OF CONTROL, ACTIVE, AND/or INACTIVE CARDS IN SUBR

SUBR *CDECK SUBR
SUBR #IFDEF,SUBR

FILECOM 1 A
FILECOM 2 A
FILECOM 3 A
FILECOM 4 A
FILECOM 5 A
FILECOM 6 A
MAIN 1 A
MAIN 2 A
MAIN 3 A
MAIN 4 A
MAIN 5 A
MAIN 6 A
MAIN 7 A
MAIN 8 A
MAIN 9 A
MAIN 10 A
MAIN 11 A
MAIN 12 A
MAIN 13 A
MAIN 14 A
MAIN 15 A
MAIN 16 A
MAIN 17 A
MAIN 18 A
MAIN 19 A
MAIN 20 A
MAIN 21 A
MAIN 22 A
MAIN 23 A
MAIN 24 A
SUBR 1 A
SUBR 2 A
LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN SUB2

Sub2
C/ LIST,NONE
Sub2
*IF DEFINIST,1
Sub2
C/ LIST,ALL
Sub2
SUBROUTINE SUB2
Sub2
*IF DEFINIUIT
Sub2
CD TITLE1
Sub2
CD SUBS1 A SAMPLE SUBROUTINE
Sub2
CD AUTHOR1
Sub2
CD Y, A, PROGRAMMER
Sub2
CD DATE WITTEN1
Sub2
CD 4-6 DATE
Sub2
*ENDIF
Sub2
*CALL DEBUG
Sub2
IF(TRACE) CALL TRACER(AMSUB2,1)
Sub2
IF(TRACE) CALL TRACER(AMSUB2,2)
Sub2
RETURN
Sub2
END

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN TRACER

TRACER
DECK TRACER
TRACER
*IF -DEF,TRACER,1
TRACER
C/ LIST,NONE
TRACER
*IF DEFINIST,1
TRACER
C/ LIST,ALL
TRACER
SUBROUTINE TRACER(SNAME,FLAG)
TRACER
*IF DEFINIUIT
TRACER
CD TITLE1
TRACER
CD TRACER STANDARD SUBROUTINE TRACE ROUTINE
TRACER
CD AUTHOR1
TRACER
CD YOUR AVERAGE PROGRAMMER
TRACER
CD DATE WITTEN1
TRACER
CD 1102179
TRACER
CD PURPOSE1
TRACER
CD TRACER PRINTS OUT TRACING INFORMATION (AT ENTRY AND EXIT TO
TRACER
CD SUBROUTINES) DEPENDING ON VALUE OF FLAG
TRACER
CD VARIABLE DICTIONARY
TRACER
CD SHAPE INPUT PARAMETER (CHAR FORMAT), CONTAINS THE SURROUNTINE
TRACER
CD NAME TO BE TRACED
TRACER
CD FLAG - INDICATOR TO TRACER OF WHAT ACTION IS TO BE DONE
TRACER
CD 1 1 ENTER SUBROUTINE SHAPE
TRACER
CD 2 2 EXIT SUBROUTINE SHAPE
TRACER
*ENDIF
TRACER
*CALL DEBUG
TRACER
IF(PARMAS) PRINT(DEHUG,*) SHAPE, FLAG
TRACER
CD 1001 CALCLULATE FLAG
TRACER
IF(FLAG,LT,1,11,FALG GT, 2) GOTO 119
TRACER
GOTO(901,302), FLAG
TRACER
901 CONTINUE
TRACER
CD ENTER SUBROUTINE
TRACER
CSTR = 1500
TRACER
WRITE(*,DEHUG,702) TEMP,7H1,111H1,SHAPE
TRACER
TRACER
LIST OF CONTROL, ACTIVE, AND/or INACTIVE CARDS IN TRACER

TRACER 999 continue

TRACER C 1 EXITSUMITUTE
TRACER IF (1SIPIR .LE. 0) TSIPIN = 1
TRACER BWRITE(901, 902) TSIPIN, TENTERED, SNAME
TRACER 702 FORMAT(1H 4=(1H), 1A4, A10)
TRACER TSIPRN = TSIPIN = 1
TRACER 1000 CONTINUE
TRACER IF (PARMS) WRITE(6, 902) SNAME, FNAME
TRACER RETURN
TRACER END

TRACER 29 = GUTU(901, 902), FLAG
TRACER 30 = WRITE(901, 902) TSIPIN, TENTERED, SNAME
TRACER 30 = WRITE(901, 902) TSIPIN, TENTERED, SNAME
TRACER 40 = END

1 AN IF STATEMENT MAY BE MORE EFFICIENT THAN A 2 OR 3 BRANCH COMPUTED GO TO STATEMENT.
TRACER 29
FE BNAME1 SYMBOIC NAME HAS TOO MANY CHARACTERS.
TRACER 33
FE .402 PRESENT USE OF THIS LABEL CONFLICTS WITH PREVIOUS USES.
TRACER 38
UNDEFINED STATEMENT NUMBERS 999

14 JUL 80 07,36,57. CERL - AIDS - MESSAGE OUTPUT PAGE 1

** WARNING DUPLICATE WRITE REQUESTED FOR COMMON BLOCK - DEBUG - REQUEST IGNORED
** TITLE A SAMPLE ADS RUN GENERATED BY THE 811H GENERATOR
** PRINT VIDE DUMP FOR ALL COMMONS
** PRINT VIDE DUMP FOR ALL KINES
** WARNING REQUESTED READ OF HEADING (ROUTINE) -- SUBJ , RECORD NOT ON MASTER FILE
** DRAW AIDE THEP
** END
TITLE: DEBUNIG VARYABLE COMMUN
VARIABLE DICTIONARY FOR COMMON BLOCK DEBUNIG

DEBUG  = DEHUG FILE NAME
PARAUS  = LOGICAL VARIABLE FOR PARAMETER TRACING
TRACE  = LOGICAL VARIABLE FOR ROUTINE TRACING CALLS TO TIMER

THE ROUTINES WHICH CALL DEBUG ARE ==
MAIN, SUB2, TRACEN

TITLE: FILECM - FILE Definitions COMMUN
VARIABLE DICTIONARY FOR COMMON BLOCK FILECM

INPUT  = ** NINE **
OUTPUT = ** NINE **

THE ROUTINES WHICH CALL FILECM ARE ==
FILED
MAIN - A SAMPLE MAIN PROGRAM
AUTHOR
T.A. PHILLIPS
DATE WITTEN
1200 A.D.

FILES USED IN MAIN ARE ==
DEBUG  INPUT  OUTPUT

VARIABLE DICTIONARY FOR ROUNTINE MAIN

PARAM1 = ** NAME **
PARAM2 = ** NAME **
PARAM3 = ** NAME **
PARAM4 = ** NAME **
PARAM5 = ** NAME **
PARAM6 = ** NAME **
PARAM7 = ** NAME **
PARAM8 = ** NAME **
PARAM9 = ** NAME **
PARAM10 = ** NAME **

THIS A MAIN PROGRAM OF LENGTH ******** WORDS.

ROUTINES CALLED BY MAIN ARE ==
SUM1
SUM2

COMPUTER BLOCKS CALLED BY MAIN ARE ==
DEBUG  FILECOMM

THE ROUTINES WHICH CALL MAIN ARE == ** NONE **

DUMP
TITLE: SUB2 A SAMPLE SUBROUTINE

AUTHOR: Y. A. PROGRAMMER

DATE WRITTEN: ANY DATE

VARIABLE DICTIONARY FOR ROUTINE SUB2  -- ** NONE **

THIS SUBROUTINE HAS A LENGTH OF ***** WORDS.

ROUTINES CALLED BY SUB2 ARF --

COMMON BLOCKS CALLED BY SUB2 ARE --

MAIN ROUTINES WHICH CALL SUB2 ARE --
TILL
SWALE = STANDARD SUBROUTINE TRACER ROUTINE

AUTHOR
YOUR AVERAGE PROGRAMMER

DATE - MITTEN
110379

PURPOSE,
TRACER PRINTS OUT TRACING INFORMATION (AT ENTRY AND EXIT TO SUBROUTINES) DEPENDING ON VALUE OF FLAG

VARIABLE DICTIONARY FOR ROUTINE TRACER:

FALG  = ** NONE **

FLAG = INDICATOR TO TRACER OF WHAT ACTION IS TO BE DONE
   = 11 ENTER SUBROUTINE NAME
   = 21 EXIT SUBROUTINE NAME

SNAME = INPUT PARAMETER (CHAR FORMAT), CONTAINS THE SUBROUTINE NAME TO BE TRACED

ISPRTN = ** NONE **

THIS SUBROUTINE HAS A LENGTH OF ........ RBUs.

ROUTINES CALLED BY TRACER ARE -- ** NONE **

COMMON BLOCKS CALLED BY TRACER ARE --

DEBUG
THE ROUTINES WHICH CALL TRACER ARE --

SUM2

IN JUL 80 073057.
THE FOLLOWING RECORDS WERE NOT FOUND ON THE MASTER FILE --

SUB1

THE ROUTINES WHICH CALL SUB1 ARE --

MAIN

I=SUB1
I_SUB2
I_TRACE
Chief of Engineers
ATTN: Tech Monitor
ATTN: DAEN-ASI-L (2)
ATTN: DAEN-CCP
ATTN: DAEN-CW
ATTN: DAEN-CNE

US Army Engineer Districts
ATTN: Library

US Army Engineer Divisions
ATTN: Library

Cold Regions Research Engineering Lab
ATTN: Library

US Government Printing Office
Receiving Section/Depository Copies (2)

Defense Technical Information Center
ATTN: DOD (12)

Engineering Societies Library
New York, NY

FES, ATTN: Library

ETL, ATTN: Library

Engr. Studies Center, ATTN: Library
Inst. for Water Res., ATTN: Library

Army Inst. and Major Activities (CONUS)

DARCOM - Dir., Inst., & Svcs.
ATTN: Facilities Engineer

AFRICOM
Aberdeen Proving Ground
Army Nails, and Mechanics Res. Ctr.
Corpus Christi Army Depot

Harry Diamond Laboratories
Dugway Proving Ground
Jefferson Proving Ground
Fort Monroe

Lackland Army Depot
Maxics Research and Dev. Ctr.
Naval Cumberland Army Depot
Pueblo Army Depot
Red River Army Depot
Rock Island Arsenal
Savannah Army Depot
Sharp Army Depot

Somers Army Depot
Tobyhanna Army Depot
Tooele Army Depot

Watervliet Arsenal
Yuma Proving Ground

White Sands Missile Range

FORSCOM
ATTN: Facilities Engineers

Fort Bragg
Fort Campbell
Fort Carson
Fort Devens
Fort Drum
Fort Hood
Fort Indiantown Gap
Fort Irwin
Fort Sam Houston
Fort Lewis
Fort McCoy
Fort McPherson
Fort George G. Meade
Fort Ord
Fort Polk
Fort Richardson
Fort Riley
Presidio of San Francisco
Fort Sheridan
Fort Stewart
Fort Wainwright
Vancouver Dts.

TRADOC
ATTN: Facilities Engineer

Fort Belvoir
Fort Benning
Fort Bliss
Carville Barracks
Fort Chaffee
Fort Dix
Fort Eustis
Fort Gordon
Fort Hamilton
Fort Jefferson
Fort Knox
Fort Leavenworth

Fort Lee
Fort McClellan
Fort Monroe
Fort Rucker
Fort Sill
Fort Leonard Wood

USACE
ATTN: Facilities Engineer

Cameron Station
Fort Leavenworth

HSC
ATTN: Facilities Engineer

Fitzsimmons Army Medical Center
Fort Carson

Walter Reed Army Medical Center

NAAC
ATTN: Facilities Engineer

Pentagon

US Army Academy
ATTN: Facilities Engineer

Culver Army Base

Bayonne MT

Suny Poly MT

US Military Academy
ATTN: Facilities Engineer

South Dakota Army Base

Brooks AFB

Ceremony Center

Chief Inst. Div., LSA, Rock Island.

US ARBCC, ATTN: Dir., Inst. & Svcs.

TACOM, Fac. Div.

TECOM, ATTN: DRSTE-LG-P

TSACOM, ATTN: STSAS-LS-ACP

DAECE, ATTN: DRBME-WK

Hq. VEFI Airborne Corps and Ft. Bragg

ATTN: AFZ-FE-EE

HQ, 7th Army Training Command

ATTN: ATETT-GEN (5)

HQ USAMC, 7th Army

ODCS (Engineer)

ATTN: AEKEN-EN (4)

V Corps

ATTN: ATYODEN (5)

VII Corps

ATTN: ATRDOD (5)

21st Support Command

ATTN: AEREN (5)

US Army Berlin

ATTN: AEBES (2)

US Army Southern European Task Force

ATTN: ASEE-ENG (9)

US Army Installation Support Activity

Europe

ATTN: NEUES-AP

US Army Japan (USARJ)

Ch. FE Div. A/JEN-FF

Fac Engm (Nanzhao)

Fac Engm (Kotanma)

ROK/US Combined Forces Command

ATTN: USA-RC/CE/Engr

416th Engineer Command

ATTN: Facilities Engineering

Norton AFB

ATTN: AFRC/EE/DEE

140
Team Distribution

Chief of Engineers
ATTN: DAEN-JSE
ATTN: DAEN-MPO-B
ATTN: DAEN-MPO-U
ATTN: DAEN-MPZ-A
ATTN: DAEN-HRD
ATTN: OAEN-MPR
ATTN: DAEN-ROL
Dept of the Army
WASH DC 20314

Chief of Engineers
ATTN: DAEN-JMS
Dept of the Army
WASH DC 20314

for forwarding to:
International Organization for Standards
Central Secretariat
1, Rue de Varembe, 1211
Geneva, Switzerland

US Army Materiel Development
and Readiness Command
ATTN: DRGDE-DK (3)
5001 Eisenhower Ave
Alexandria, VA 22333

Commander
US Army Electronics Command
ATTN: DRSEL-TL-M/Mr. Tenzer
Fort Monmouth, NJ 07703

DOD ADP Policy Committee
Assistant Secretary of Defense (Controller)
WASH DC 20301

DOD Standardization Area
Computer Aided Design and Numerical Control
Naval Ship Engineering Center
Hyattsville, MD 20782

DOD Standardization Program
for Information Processing Standards for Computers (IPSC)
Directorate of Data Automation (AF/KRAX)
HQ, USAF
WASH DC 20330

Commander
7th Army Combined Arms Training Center
ATTN: AETTM-HRD-END
APO New York 09407

US Army Engr Div, Europe
ATTN: Technical Library (3)
APO New York 09767

American National Standards Institute
1430 Broadway
New York, NY 10018

ANSI X3 Committee
C/O CBEMA
1828 L Street, NW
WASH DC 20036

DOD Working Group on Computer Documentation Standards
DOD/DCON
The Pentagon
WASH DC 20301

DOD Working Group of Computer-Generated Military Symbology
DOD/DISPLY
The Pentagon
WASH DC 20301

Federal Information Processing Standards Coordinating and Advisory Committee
Dept of Commerce
WASH DC 20234

Library of Congress
Interagency Committee on Automatic Data Processing
National Bureau of Standards
WASH DC 20234

National Bureau of Standards
Institute for Computer Sciences and Technology
WASH DC 20234
DAI FILM