XPEDITER/TSO and XPEDITER/IMS

COBOL User's Guide

Release 7.0

COMPUWARE
®

Please direct questions about XPEDITER/TSO and XPEDITER/IMS
or comments on this document to:

XPEDITER/TSO and XPEDITER/IMS Technical Support
Compuware Corporation
31440 Northwestern Highway
Farmington Hills, MI 48334-2564

1-800-538-7822

Outside the USA and Canada, please contact
your local Compuware office or agent.

Doc. CWXTUCX7A

January 24, 2001.

Contents

Figures

Tables

Preface

XPEDITER is a testing and debugging tool for COBOL, Assembler, PL/I, and C programs that run in TSO (both interactive and batch), IMS, CICS, and ROSCOE environments. XPEDITER simplifies the tasks of testing and debugging by giving programmers complete control over the execution of their programs at the source code level without requiring source or object code changes.

Members of Compuware's XPEDITER family of debugging and testing products include XPEDITER/TSO, XPEDITER/IMS, XPEDITER/CICS, XPEDITER/Code Coverage, XPEDITER-/ROS, XPEDITER/Xchange, and XPEDITER/DevEnterprise (for the workstation).

Summary of Changes

XPEDITER/TSO and XPEDITER/IMS Release 7.0 includes the following enhancements:

- DB2 Stored Procedure Support

  XPEDITER now provides the user an ability to debug DB2 Stored Procedures under the control of DB2 and Workload Manager (WLM).

- IBM C Language Support

  IBM C Language is now available for all clients. XPEDITER provides source level debugging of C Language programs.

- XPEDITER/TSO and File-AID Integration

  XPEDITER has added the option to call File-AID directly from the XPEDITER Primary menu. The user can now issue FADB2 within a test to invoke File-AID directly.

- Compile Facility Support for C Language Compiles

  The Compile Facility now provides options to select and compile IBM C Language programs.

- High Level Assembler Enhancements

  XPEDITER has been enhanced to recognize the latest OP-CODES. Mixed case support has also been added.

- IEEE Floating Point Support

  XPEDITER provides support for use of the new IEEE Floating Point registers. Commands have also been added to allow the user to display those values stored in the Floating Point registers.

- Access Register Protection

  XPEDITER has added support to protect the values in the user's Access Registers. Some system functions overlay the register and require protection for the user. For user clarification, it should also be noted that XPEDITER does not yet support the Access Register execution mode.

- Code Coverage Enhancements

  Users may now assign a TESTID to the Code Coverage statistics which are collected. A USERID will automatically be collected by XPEDITER. XPEDITER/Code Coverage provides the ability to report on these new fields. XPEDITER/TSO also allows the unexecuted code from a previous Code Coverage session to be ***Highlighted*** in an XPEDITER/TSO debugging session.

- PROFILE Command From the Test Panels

  Prior to invoking a test under XPEDITER, the user may now issue the PROFILE command in order to change the currently selected test profile. This will allow a faster switching of test profiles.

- Batch Connect JCL Enhancement

  Support for the Included and Nested procedures has been added in XPEDITER/TSO Release 7.0.

- Addition of the EXIT ABEND Command

  The EXIT ABEND command will allow the user to end the test session with an Abend.

In conjunction with the XPEDITER/TSO and XPEDITER/IMS 7.0 Release Announcement, the following changes to XPEDITER/TSO and XPEDITER/IMS environment support are now defined for Release 7.0. Only revisions to the supported environments are displayed below:

- MVS/ESA Versions 4.1 and above
- VS COBOL II Version 1, Releases 4.0 and above
- PL/I Versions 2.3 and above
- ISPF/PDF Versions 3.5, 4.1, 4.2, 4.3, 4.4, and 4.5
- DB2 Versions 3.1, 4.1, 5.1, and 6.1
- IMS/ESA Versions 4.1, 5.1, 6.1, and 7.1
- C for MVS/ESA Version 3.2
- OS/390 C Version 1.2, 1.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, and 2.10

Note: Contact Compuware Technical Support for information on Releases not mentioned in the “Summary of Changes”.

Introduction

This document introduces you to XPEDITER/TSO and XPEDITER/IMS, using a series of examples that demonstrate simple and advanced debugging techniques that can be applied to your application development and maintenance needs. It describes how to use XPEDITER/TSO facilities in the COBOL context with minor references to mixed language debugging support. Additional information is provided in other manuals; refer to "Related Publications" on page -xxii.

Manual Organization

This manual contains the following chapters:

Chapter 1. XPEDITER/TSO Overview: Summarizes the functions and features of XPEDITER/TSO and XPEDITER/IMS.

Chapter 2. User Interface to XPEDITER/TSO: Describes the Source screen format, default PF keys, command processing, and attention key processing.

Chapter 3. Quick Sample Debugging Session: Provides a short tutorial highlighting the basic debugging features of XPEDITER/TSO.

Chapter 4. Getting Started: Describes the online Compile Facility, which is used to prepare your programs for debugging with XPEDITER/TSO. Also describes how to start an XPEDITER/TSO debugging session in interactive mode, in batch mode using the batch connect facility, and how to manually change your execution JCL to start a batch session.

Chapter 5. Debugging Interactively: Describes how to use XPEDITER/TSO commands to control execution, inspect data, analyze logic, modify logic, debug sourceless programs, and other debugging functions.

Chapter 6. Handling Run-Time Errors: Describes how to analyze and apply fixes for abends.

Chapter 7. Debugging With XPEDITER/IMS: Describes how to start an XPEDITER/IMS debugging session.

Chapter 8. Debugging With DB2 Stored Procedures: Describes how to start a debugging session with DB2 Stored Procedures.

Chapter 9. Debugging Programs With Special Conditions: Provides general information about debugging certain types of programs.

Appendix A. Using the File Allocation Utility: Describes how to access and use the File Allocation Utility.

Appendix B. XPEDITER/TSO Environment Test Screens: Provides full descriptions of all environment test screens.

Appendix C. Specifying Setup Options: Describes how to specify and override the installed setup options such as the load library, DDIO file, DB2 system name, and so on.

Appendix D. Specifying Session Defaults: Describes how to specify terminal characteristics, PF key definitions, screen colors, user profiles, and so on.

Appendix E. XPEDITER/TSO Utilities: Describes the XPEDITER/TSO utilities that let you create and maintain source listings, display available memory, display file allocations, and convert and merge user profiles.

Appendix F. Binding the Application Plan: Describes how to bind DB2 programs.

Appendix G. Double Byte Character Set (DBCS) Support: Describes the complexity of double-byte character set support.

Intended Audience

This manual is intended for use by COBOL programmers. A working knowledge of COBOL and your local operating system(s) is assumed.

Accessing Other Products

XPEDITER/TSO and XPEDITER/IMS are fully integrated with Compuware's File-AID for DB2 (Release 3.5 or above), Abend-AID (Release 7.0.2 or above for COBOL and Assembler, and Release 8.0.4 or above for PL/I and C Language), and XPEDITER/DevEnterprise Batch Debugger (Release 2.0). From the XPEDITER/TSO Primary Menu, you can access a CICS region where other Compuware tools, such as XPEDITER/CICS and CICS Abend-AID, can be used.

Notation Rules

This manual uses the following notation rules:

- Uppercase characters are used in text for commands, keywords, and function keys.
- Screen titles appear in text with initial capitals.
- Screen fields, prompts, and messages are capitalized as displayed.
- Information that the user enters, including the *Enter* command itself, is normally shown in **boldface** type.
- Occasional dataset and library member names are shown in **boldface** type for emphasis.
- Commands are not considered case sensitive; they can be typed in all lowercase in the command area.

Related Publications

The following Compuware documents are also available for use with the XPEDITER/TSO and XPEDITER/IMS product:

- *XPEDITER/TSO and XPEDITER/IMS Installation Guide*

  This document contains the steps necessary to install XPEDITER/TSO and XPEDITER/IMS (full-screen IMS/DC) for testing COBOL, Assembler, PL/I, and C Language programs.

- *XPEDITER/TSO and XPEDITER/IMS Reference Manual*

  This document lists all the XPEDITER commands available for use when debugging COBOL, Assembler, PL/I, and C Language programs. It is written for the application programmer engaged in program development and maintenance.

- *XPEDITER/TSO and XPEDITER/IMS Assembler User's Guide*

  This document contains a series of examples that demonstrate simple and advanced debugging techniques that can be applied to your application and development needs. It describes how to use XPEDITER/TSO and XPEDITER/IMS facilities in the Assembler context.

- *XPEDITER/TSO and XPEDITER/IMS C User's Guid*e

  This document contains a series of examples that demonstrate simple and advanced debugging techniques that can be applied to your application and development needs. It describes how to use XPEDITER/TSO and XPEDITER/IMS facilities in the C Language context.

- *XPEDITER/TSO and XPEDITER/IMS Messages and Codes*

  This document provides messages and codes for XPEDITER/TSO and XPEDITER/IMS.

- *XPEDITER/TSO and XPEDITER/IMS PL/I User's Guide*

  This document contains a series of examples that demonstrate simple and advanced debugging techniques that can be applied to your application and development needs. It describes how to use XPEDITER/TSO and XPEDITER/IMS facilities in the PL/I context.

- *XPEDITER/TSO and XPEDITER/IMS Quick Reference*

  This document provides quick access to XPEDITER/TSO COBOL, Assembler, PL/I, and C Language commands and command descriptions.

- *File-AID for DB2 Reference Manual*

  This document provides information about using File-AID for DB2 facilities.

- *XPEDITER/DevEnterprise Suite of Manuals*

  These documents provide information about using XPEDITER/DevEnterprise on the workstation.

- *XPEDITER/Xchange Installation and Reference Manual*

  This document provides the information needed to move your time-sensitive applications into the year 2000 and beyond.

- *Enterprise Common Components Installation and Customization Guide*

  An introduction, overview, and installation guide for Compuware Shared Services components, Compuware's common files and utilities (DDIO and DDSUTIL), and the Compuware language processor.

- *Compuware Shared Services User/Reference Guide, MVS Version*

  This document provides information to install, customize, and maintain Compuware Shared Services (CSS). It is intended for CSS installers and application programmers.

- *XPEDITER/Code Coverage Mainframe Installation Guide*

  This document provides information about installing XPEDITER/Code Coverage.

- *XPEDITER/Code Coverage Mainframe User/Reference Guide*

  This document provides information regarding the use of XPEDITER/Code Coverage. It provides an explanation of the requirement that XPEDITER/Code Coverage work in conjunction with at least one additional mainframe XPEDITER test and debugging tool.

More information on compiler options is contained in the following documents:

- *VS COBOL II Application Programming Guide*
- *IBM OS/VS COBOL Compiler and Library Programmer's Guide*
- *IBM OS Full American National Standard COBOL Compiler and Library, Version 3 Programmer's Guide*

Documentation Availability

FrontLine Support Web Site

You can access online technical support for Compuware products via our FrontLine support Web site. You can read or download documentation, frequently asked questions, and product fixes, or directly e-mail Compuware with questions or comments. To access FrontLine, you must first register and obtain a password at **http://frontline.compuware.com**.

Online Documentation

Documentation for this product is provided on CD-ROM in several electronic formats. PDF files can be viewed with the free Adobe Acrobat Reader, available at **http://www.adobe.com**. HTML files can be viewed with any standard web browser.

World Wide Web

Compuware's site on the World Wide Web provides information about Compuware and its products. The address is **http://www.compuware.com**.

Getting Help

At Compuware, we strive to make our products and documentation the best in the industry. Feedback from our customers helps us to maintain our quality standards.

Questions about any XPEDITER product or comments on this document should be directed to:

XPEDITER/TSO and XPEDITER/IMS Technical Support

Compuware Corporation
31440 Northwestern Highway
Farmington Hills, MI 48334-2564

1-800-538-7822

If problems occur, consult your manual or the XPEDITER/TSO and XPEDITER/IMS technical support representative at your site. If problems persist, please obtain the following information before calling Compuware. This information helps us to efficiently determine the cause of the problem.

For problems occurring during compile time:

- Release level of COBOL
- List of other vendor products used
- Abending module name
- Any messages in CWPERRM ddname

For problems encountered before bringing up your program:

- Sequence of events leading up to the problem
- Release and level of the XPEDITER product
- Any ISPF error messages, operating system messages, and information provided by the TSO Command Profile WTPMSG

For run-time abends:

- Release and level of the XPEDITER product

- Any ISPF error messages, operating system messages, and information provided in the TSO Profile WTPMSG
- Information provided in the product log dataset at the time of the abend

Note: When in-depth diagnosis is required, the following will be requested:

1. Full SYSUDUMP (not Abend-AID dump) or SYSMDUMP
2. Source listing or tape of problem programs, including all copy books and members used
3. Listing of the XPEDITER log at the time of the abend
4. Listing of the original compile
5. Listing of the BTSOUT dataset, if testing under BTS.

Chapter 1. XPEDITER/TSO Overview

XPEDITER/TSO and XPEDITER/IMS are debugging and testing tools for COBOL, Assembler, PL/I, and C programs. XPEDITER automates the tasks of identifying problems, applying solutions, analyzing the impact of changes, and testing the fixes.

Compiling and assembling your programs, test session setup procedures, and XPEDITER commands are compatible across languages, with some differences.

When you have COBOL, Assembler, PL/I, and/or C language versions installed at your site, XPEDITER/TSO allows you to debug mixed applications in the same test session.

Operating Environment Support

XPEDITER/TSO is invoked as an ISPF dialog under TSO. XPEDITER/IMS DC executes in an IMS-dependent region within your TSO address space. The following environments are supported:

- MVS/ESA Versions 4.1 and above
- OS/390 Releases 1, 2, 3, 4, 5, and 6
- DFSMS
- OS/VS COBOL Release 2.4
- VS COBOL II Version 1, Releases 4.0 and above
- COBOL for MVS & VM Release 1.2 (in compatibility mode)
- COBOL for OS/390 Release 2.1 (in compatibility mode)
- CA-OPTIMIZER Releases 5.1 and 6.0
- Assembler H Version 2, High Level Assembler
- PL/I Versions 2.3 and above
- AD/CYCLE for PL/I MVS and VM 1.1 (PL/I 370)
- Major sorting packages - DFSORT and SYNCSORT
- ISPF/PDF Versions 3.5, 4.1, 4.2, 4.3, 4.4, and 4.5
- DB2 Versions 3.1, 4.1, 5.1, and 6.1
- IMS/VS Versions 2.1 and 2.2
- IMS/ESA Versions 4.1, 5.1, 6.1, and 7.1
- BTS
- IDMS/DB, ADABAS, TOTAL, TIS, SUPRA, DATACOM/DB, System 2000
- Hogan
- AD/CYCLE for COBOL/370 Release 1.1
- LE/370 Versions 1.2, 1.3, and 1.4
- Language Environment for MVS & VM, Versions 1.5, 1.6, 1.7, and 1.8.
- C for MVS/ESA Version 3.2
- OS/390 C Version 1.2, 1.3, 2.4, 2.5, 2.6, 2.7, and 2.8

Note: Contact Compuware Technical Support for information on Releases not mentioned in the "Operating Environment Support".

Notes:

1. COBOL for MVS & VM Version 1.2 compatibility mode means that XPEDITER/TSO supports COBOL II and COBOL/370 programs that have been recompiled with COBOL for MVS & VM Version 1.2. XPEDITER/TSO does not support the new COBOL for MVS & VM Version 1.2 features, with the exception of Local-Storage, Recursive, and Returning (requires Compuware Shared Services (CSS) Release 7.4 or greater).

2. In a prior release, support was dropped for Assembler F; VS COBOL II, Releases 3.0, 3.1, and 3.2; PL/I Version 1.5; ISPF/PDF Versions 2, 3.1, 3.2 and 3.3; DB2 Versions 1 and 2; IMS/ESA Version 3; MVS/XA Version 2; IMS/VS Versions 1 and 2; CA-OPTIMIZER Release 5.0; TSO dBUG-AID; and TCF.
3. All release support is dependent on the vendor certifying that the product is Y2K compliant.

Modes of Operation

There are two modes in which an XPEDITER/TSO debugging session can be invoked:

- Interactive mode
- Batch mode

Note: XPEDITER/IMS can only be executed in interactive mode.

Interactive Mode

In this mode, you can interactively allocate the necessary files and databases required for the program to execute, set up the test session environment options, and optionally override any XPEDITER defaults. As you watch your program execute, you can use execute and you can use interactive source-level debugging functions such as stepping through the source, setting breakpoints, displaying and modifying variables, etc..

Interactive mode is preferable when you have a specific problem for which you want to dynamically try out a solution.

Batch Mode

There are two ways to invoke an XPEDITER/TSO debugging session in batch mode:

- Batch connect facility.

  Through the batch connect facility, your execution JCL is automatically submitted to MVS and you can connect directly to the job as it executes in its native environment—the MVS batch initiator.

  The batch connect facility automates the setup and file allocations, displays the job steps in the specified JCL, and lets you select the steps to which you want to interactively connect and those that you want to run in unattended batch. The batch connect intelligent scanner automatically expands the JCL and inserts the necessary statements to run each step according to how you want it to be processed.

  With the batch connect facility, you get the benefit of submitting the job in batch, which uses less processor resources, and you can test programs with multiple steps, programs that require tape files and many I/O operations in background, and long running programs.

- Manually changing your execution JCL.

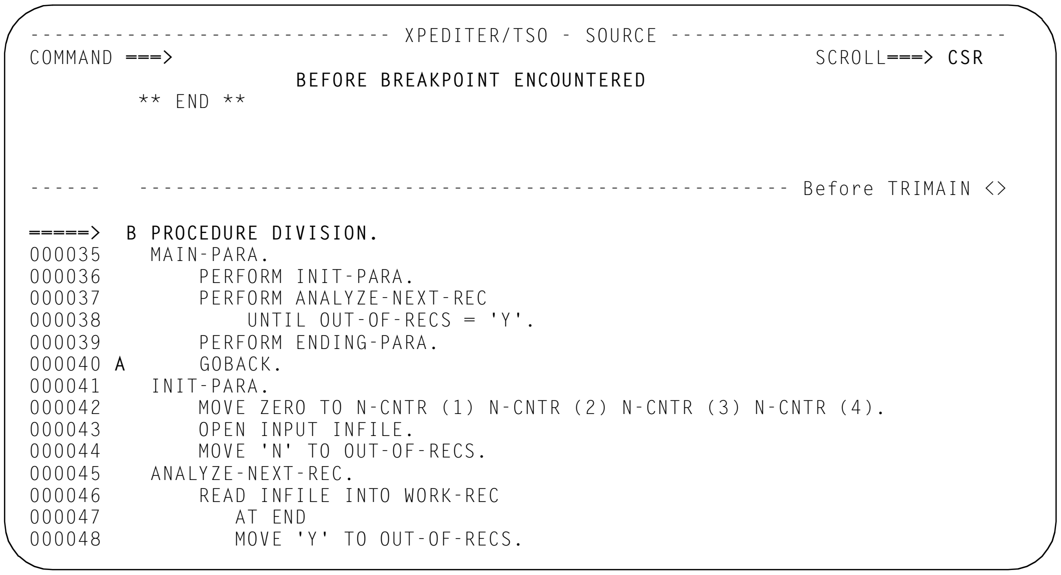
  Although the batch connect facility is the preferred method for running a job in batch, you can manually access your execution JCL and make changes to convert the JCL to run with XPEDITER/TSO in unattended or interactive (batch connect) modes.

Interactive Debugging and Testing Features

When your program is executed under interactive XPEDITER/TSO, the source is displayed in a fully scrollable window where you can view the inner workings of the program as each statement is executed. For example, you can see the execution arrow moving from a PERFORM statement to the actual out-of-line perform paragraph, or you can see a data identifier changing its value as the MOVE, INITIALIZE, STRING, or COMPUTE statements are executed.

Figure 1-1 on page 1-3 shows the Source display screen under XPEDITER/TSO. All debugging and testing functions are accessible from this screen without having to exit to a separate screen.

XPEDITER/TSO does not require any source code or load module changes; however, you need to create a "symbolic" information dataset called the source listing file to debug at the source level. The source listing is created by compiling your programs with the Compuware Shared Services COBOL language processor. XPEDITER/TSO displays and processes the source listing records during the test session.



```
----------------------------- XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                      SCROLL===> CSR
                      BEFORE BREAKPOINT ENCOUNTERED
         ** END **



------   ---------------------------------------------------- Before TRIMAIN <>

=====>  B PROCEDURE DIVISION.
000035    MAIN-PARA.
000036        PERFORM INIT-PARA.
000037        PERFORM ANALYZE-NEXT-REC
000038            UNTIL OUT-OF-RECS = 'Y'.
000039        PERFORM ENDING-PARA.
000040 A      GOBACK.
000041    INIT-PARA.
000042        MOVE ZERO TO N-CNTR (1) N-CNTR (2) N-CNTR (3) N-CNTR (4).
000043        OPEN INPUT INFILE.
000044        MOVE 'N' TO OUT-OF-RECS.
000045    ANALYZE-NEXT-REC.
000046        READ INFILE INTO WORK-REC
000047           AT END
000048           MOVE 'Y' TO OUT-OF-RECS.
```

Figure 1-1. COBOL Program in the Source Display Screen

The capabilities of XPEDITER/TSO are extensive enough to assist the experienced programmers' needs, and yet, the implementation is simple enough and efficient enough to curtail the learning curve that novice programmers generally experience. The following list summarizes XPEDITER/TSO's major features:

- Intercept program abends.

  XPEDITER/TSO detects application abends and displays a diagnostic message. You can request an Abend-AID formatted report if you have Abend-AID release 7.0.2 or above. You can also view the log for more information. If the problem is a recoverable error, you can fix the error dynamically and resume execution.

- Start or stop execution at any point.

  You can set conditional or unconditional breakpoints, so you can stop program execution at any point and figure out what the program has processed so far, and what the consequences of any changes could be. You can also step through code line-by-line to understand how each statement affects the program.

- Display and modify variable, register, and storage contents.

  You can view the values of any variable as your program executes, and if desired, alter the values as if the program had actually moved in the value you requested. Tables can be displayed by dimension, and you can browse through each entry by incrementing and decrementing the index or subscript.General-purpose registers and memory can also be displayed to analyze the problem at the lower level.

- Trace logic flow.

Statements can be highlighted to identify the execution path as XPEDITER/TSO traces through the program. You can control the speed of the execution so that you can follow the logic interactively at a speed that best suits you.

- Logically review execution in the reverse direction.

  XPEDITER/TSO can display the execution history in the reverse direction so you can review the execution path with associated data values. The execution arrow moves backwards, and the data display shows the original values.

- Monitor execution coverage.

  You can set counters on statements to test execution coverage or to monitor loops and procedure calls for optimization.

- Alter logic.

  You can dynamically change the control flow by modifying the data values or by forcing a branch to test an alternate path.

- Bypass unwanted code.

  You can skip a range of statements or programs.

- Temporarily insert XPEDITER/TSO debugging statements.

  You can insert XPEDITER/TSO commands to control debugging conditions that occur at a certain location and to force data and logic changes. This feature lets you prototype some COBOL-like constructs. If you have XPEDITER for DB2 Extension and File-AID for DB2, you can also insert SQL statements.

- Analyze program structure and data flow.

  XPEDITER/TSO identifies certain program structures (conditionals, branches, I/Os, statements that alter data values, etc.) by highlighting the applicable statements when requested. Also, statements that make references (MODIFY or USE) to a particular data item can be identified.

- Display file status and DCBs.

  VSAM file status and last I/O operation can be retrieved through XPEDITER/TSO. DCB information can also be checked in the case of I/O error. When a DD statement is not preallocated, XPEDITER/TSO intercepts execution and gives you an opportunity to allocate the missing files before resuming execution.

- Test and debug programs that do not have the source available.
- Display called module stacking.

  The called module configuration such as the load address, entry point, size, attribute, AMODE, RMODE, and language can be retrieved.

- Maintain multiple profiles.

  XPEDITER/TSO’s Profile Handling facility provides a flexible and easy-to-use method for using and maintaining multiple profiles. Multiple profiles are used because some of the data you see during an XPEDITER/TSO debugging session depends on the current profile. This includes installation defaults established by your system programming staff, and most importantly, environment parameters and setup options for the debugging session.

XPEDITER/TSO Input and Output

The primary input to XPEDITER/TSO is the following:

1. DDIO libraries containing the source listing members.
2. Load libraries containing the programs to be tested and debugged.

Note: The source listing dataset and your load libraries are the output of the compile and link process discussed in "Preparing Your Programs" on page 4-3.

3. JCL, file list, or CLIST that can be processed to allocate the input files and databases needed by your program.
4. Optionally, a test script library.

The primary output of the debugging session is the following:

1. Data files generated by the execution of the program.
2. Session log.

   XPEDITER/TSO automatically records (to the log file) the commands entered and the responses made to each command during a debugging session. The log dataset can have an LRECL of 133 or 80. The log does not contain the interactive manipulation commands (e.g., LEFT, RIGHT, UP, DOWN) nor does it contain the actual program displays.

   In *unattended batch mode*, the log file shows the output of a batch test and is available after the completion of the debugging session. In *interactive mode*, it is available at all times.

   The log file can be kept as part of the documentation, to be referenced whenever maintenance is performed on the program. It can be viewed for further information in determining the cause and possible resolution of an abend. "Handling Run-Time Errors" contains a discussion of the log display.
3. Test script containing the commands that were entered during the test session.

   After the session is terminated, you can save, copy, or move the generated script to a member of a script library (INCLUDE dataset), which can then be used as input to another debugging session.

Types of Programs Supported

XPEDITER/TSO supports the following types of programs:

- Standard batch applications
- ISPF dialog applications
- Batch applications that issue database calls (IDMS, IMS/DB, DB2, IDMS/DB, ADABAS, and SUPRA)
- IMS/DC programs using BTS
- Hogan applications (BATCHPEM, DLIPEM, BMPPEM, and IMSPEM)

XPEDITER/IMS supports the following types of programs:

- IMS/DC programs executing in the IMS message, BMP, or Fast Path region
- Hogan applications executing in the IMS message region

LOGON Region Size Requirements

Debugging with XPEDITER/TSO may require an increase to the default TSO LOGON region size. This provides protection against abends caused by insufficient space.

XPEDITER/TSO requires about 640K bytes minimum, in addition to the application programs to be tested and debugged. Additional storage could be required, depending on the number and size of the programs that are referenced symbolically during the debugging session.

It has been our experience that 2048K to 4096K (2M to 4M) is sufficient size for debugging most applications.

Restrictions and Warnings

The following list explains some of the technical restrictions and questions related to the functioning of XPEDITER/TSO:

Compiler Options:

XPEDITER requires specific compile options which may vary depending on the language type and actual version being used. A user should always refer to the appropriate *CSS User/Reference Guide* for information related to any required compiler options. Examples are as follows: (1) Options NOTEST and NONUMBER are required for COBOL; (2) Option NOTEST is required for PL/I; and (3) Options ESD and LIST are required for Assembler. The *guide* is required to assist users in manually converting Compile and Link JCL to execute the CSS Language Processor. The *guide* also gives most users information on how to set up the Language Processor parameters to write a matching source listing to a DDIO file.

Postprocessed Listing Support:

When the postprocessed listing contains a COPY SUPPRESS statement, the XPEDITER/TSO commands FIND, PEEK, and KEEP will not work correctly.

The postprocessed source listing must have been compiled with the compile options required by XPEDITER/TSO; otherwise, unpredictable results can occur.

Attention (User Interrupt—PA1 Key) Processing:

The use of the PA1 key (user interrupt) from within XPEDITER/TSO can sometimes cause a recursive abend.

Split Screen:

When running XPEDITER with the LE/370 run-time libraries with TRAP ON, **do not** split the screen and run another application that uses the LE/370 libraries. An abend may occur.

Multitasking Support:

Compuware does not support multitasking applications within XPEDITER/TSO, except in the dialog environment, where single task testing is supported within a multitasking environment.

Self-Modifying Programs:

XPEDITER's ability to debug an application program is based on its inherent knowledge of the object module. Self-modifying code that changes opcodes, displacements, or operands is not supported for testing under XPEDITER.

Optimized Code:

XPEDITER/TSO displays the source for optimized code as it was originally written, but executes the code generated by the optimizer. As a result, depending on the optimizing algorithm applied to the code, the following can occur:

- Highlighting during execution of the trace of the optimized code can be misleading.
- Values displayed by the KEEP and PEEK commands may not be updated according to the program logic.

- Abends can occur when you use XPEDITER/TSO commands such as GOTO and SKIP that alter the program execution paths. These abends occur when the altered execution paths are in conflict with path dependencies generated by the optimizer.
- If you used Copy Suppress in the Procedure Division, you might not be able to set a breakpoint on the first statement following the copied code.

IMS Testing:

- When an application program is scheduled into an IMS MPP or BTS simulated MPP region, none of the breakpoints are retained from a previous test. Once the application program is scheduled, the breakpoints are retained until the application program returns to the IMS program controller.
- Transactions specified on the test MPP screen are queued to run in the XPEDITER/TSO region regardless of where the transactions were initiated. XPEDITER/TSO assigns a unique class to each of the specified transaction codes and forces them to run only in your XPEDITER/TSO region. The transaction codes are not reassigned to their original class until you have completely finished testing in the MPP region.
- If a DLI program uses the XRST facility, the first DLI call **must** be the XRST call. Otherwise, you can cause an abend 04E with reason code 00D44054.
- Within a BTS/DLI setup, recovery of DB2 tables and IMS databases are uncoordinated. An SQL COMMIT/ROLLBACK call commits or rolls back changes made to DB2 tables only, **not** your IMS databases.
- XPEDITER/TSO forces on the PARDLI parameter in the BMP environment.
- If you are experiencing system abends while using the BTS TRACE to monitor DB2 activity, check with IBM for fixes available for the BTS program product. Problems have occurred in MVS/XA IMS 1.3 and BTS 1.3, as well as MVS/ESA IMS 3.1 and BTS 3.1.

Naming Convention:

Program names should not begin with IBM reserved prefixes such as IHO, ILBO, IGZ, and so on.

COBOL Ready Trace:

Unpredictable results can occur in XPEDITER/TSO when the COBOL READY TRACE command is used.

Memory Allocations:

User programs and other vendor packages that perform FREEMAINs on subpool=0 or subpool=3 are not supported.

Link Options:

XPEDITER/TSO does not support the NE or OVLY linkage editor parameters.

Abend Intercepts:

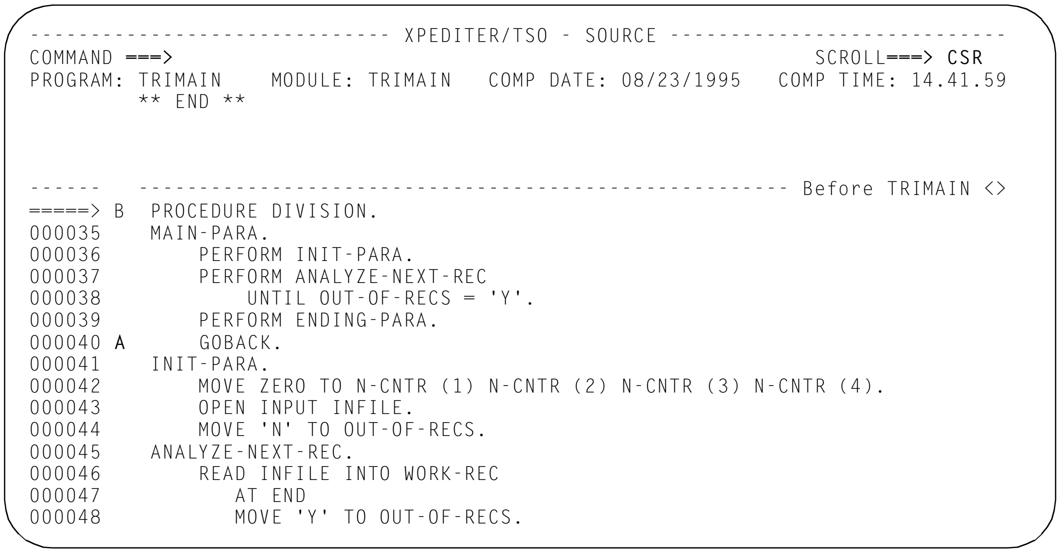
XPEDITER/TSO does not support programs that issue (E)STAE or (E)SPIE macros.

Chapter 2. User Interface to XPEDITER/TSO

This chapter describes how to interact with XPEDITER/TSO in general. The screen formats, PF key assignments, command processing, and attention key processing are discussed.

XPEDITER Screens

XPEDITER uses screens that are like ISPF/PDF, making the XPEDITER menus and utility screens self-explanatory. The Source, Log, Show, and Memory screens have a similar format. Figure 2-1 displays a COBOL program in the XPEDITER Source screen.



```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                   SCROLL===> CSR
PROGRAM: TRIMAIN    MODULE: TRIMAIN   COMP DATE: 08/23/1995   COMP TIME: 14.41.59
         ** END **



------   ------------------------------------------------------ Before TRIMAIN <>
=====> B  PROCEDURE DIVISION.
000035    MAIN-PARA.
000036        PERFORM INIT-PARA.
000037        PERFORM ANALYZE-NEXT-REC
000038            UNTIL OUT-OF-RECS = 'Y'.
000039        PERFORM ENDING-PARA.
000040 A      GOBACK.
000041    INIT-PARA.
000042        MOVE ZERO TO N-CNTR (1) N-CNTR (2) N-CNTR (3) N-CNTR (4).
000043        OPEN INPUT INFILE.
000044        MOVE 'N' TO OUT-OF-RECS.
000045    ANALYZE-NEXT-REC.
000046        READ INFILE INTO WORK-REC
000047           AT END
000048           MOVE 'Y' TO OUT-OF-RECS.
```

Figure 2-1. XPEDITER/TSO COBOL Program in the Source Screen

The screen areas are described as:

Title
(line 1)—Identifies the screen name: Source, Log, Show, Memory, and so forth.

Command area
(line 2)—Primary command line, which can be increased to three lines using the SET CMDSIZE command.

Scroll amount
(line 2)—Indicates the current scroll amount. You can type over the current value with one of the following values:

1 to 9999	Scrolls by the number of lines or columns.
CSR or C	Scrolls based on the current position of the cursor.
DATA or D	Scrolls by one line or column less than PAGE.
HALF or H	Scrolls by a half page.
PAGE or P	Scrolls by one page.

Program
(line 3)—Identifies the source program currently displayed. This is an unprotected field and can be typed over with another program name.

Module
(line 3)—Displays the load module name.

Compile date
(line 3)—Displays the compile date.

Compile time
(line 3)—Displays the compile time.

Message area
(line 3)—Displays short or informational messages. When a message is issued, it overlays the program information in line 3. Press **Enter** to flush the message and display the program information. Additional information can be accessed by pressing **PF1** (HELP).

Keep window
(lines 4 - 8)—Automatically displays data referenced in the current statement; i.e., the statement where the execution arrow is located when the breakpoint takes effect. Explicitly kept data is also displayed. Explicitly kept items are denoted by a K in column 9 of the window.

The data in the window can be scrolled by moving the cursor to the Keep window and using the PF7 (UP) and PF8 (DOWN) keys to scroll vertically and the PF22 (DRIGHT) and PF23 (DLEFT) keys to scroll horizontally. You can also control the size of the window and, for automatic keeps, the placement of the automatically kept items. Refer to the SET command in the *XPEDITER/TSO and XPEDITER/IMS COBOL Reference Manual* for additional information.

The SET AUTOKEEP ON/OFF command toggles the effect of the Automatic Keep function.

Execution status
(line 9)—Identifies the current execution point in your program. The <> shown at the end of this line indicates that the source can be scrolled to the left and/or right.

Source area
As shown in Figure 2-1 on page 2-1, the source area begins on line 10 and displays 68 to 70 bytes of the source code on the screen at a time. The source can be scrolled vertically using the PF7 (UP) and PF8 (DOWN) keys and horizontally using the PF10 (RIGHT) and PF11 (LEFT) keys. When a Peek window is visible, the data in the window can also be scrolled vertically using the PF7 and PF8 keys and horizontally using the PF22 and PF23 keys.

The After, Before, Peek, and Skip indicators are displayed on the left side of the source in column 9. A 7-digit counter set by the COUNT command is displayed on the right side beginning at column 74.

PF Keys

The default settings for the 24 XPEDITER PF keys are listed below. These values are valid during the XPEDITER/TSO session. ISPF PF keys are not affected. The ISPF KEYS command or the SET PF*n* command can be used to override the defaults.

PF Key	Default Setting	Description of Function
PF1/PF13	HELP	Elaborates an XPEDITER/TSO message and invokes the context-sensitive tutorial facility.
PF2	PEEK CSR	Displays the contents of the data name defined by the current cursor position. The cursor must be in the Source window under a valid data name.

Table 2-1. (Page 1 of 2) Default Program Function (PF) Keys.

PF Key	Default Setting	Description of Function
PF14	FIND CSR	Finds the character string located under the cursor position.
PF3/PF15	END	Returns you to the previous menu if you are in the Log, Help, Browse, or Show functions.
PF4/PF16	EXIT	Ends the current XPEDITER/TSO session.
PF5	FIND	Repeats the action of the previous FIND command.
PF17	FIND IND	Scrolls the source display to successive levels of indirect references related to a previously entered FIND INDIRECT command.
PF6/PF18	LOCATE *	Scrolls the source display to the current location where execution was suspended.
PF7/PF19	UP	Scrolls the source or data in the Keep window up, or toward the top of the file.
PF8/PF20	DOWN	Scrolls the source or data in the Keep window down, or toward the bottom of the file.
PF9/PF21	GO 1	Executes the next logical instruction in your program, then pauses.
PF10	LEFT	Scrolls the source display to the left.
PF11	RIGHT	Scrolls the source display to the right.
PF12/PF24	GO	Starts or resumes execution of your program.
PF22	DLEFT	Scrolls data in an Automatic Keep, Keep, or Peek window to the left.
PF23	DRIGHT	Scrolls data in an Automatic Keep, Keep, or Peek window to the right.

Table 2-1. (Page 2 of 2) Default Program Function (PF) Keys.

Command Processing

In interactive mode, the results of command execution are immediately visible on the source display.

XPEDITER/TSO commands can be entered in three ways:

1. Type the command in the primary command area and press **Enter**. Command stacking, delimited by a semicolon (;), is allowed.

   The primary command area can be extended up to three lines by using the SET CMDSIZE command. The previous primary command can be recalled by entering a question mark (?).

2. Press the PF key that was assigned to the desired command. Refer to Table 2-1 on page 2-2 for a list of the PF key assignments.

3. Type over the 6-digit compiler-generated statement number with a valid line command and press **Enter**.

   XPEDITER/TSO records the line command in the log in the same manner as the primary command.

By default, commands entered in lowercase are converted to uppercase. To override the default, use the SET CAPS OFF command. Also, to display lowercase data, use the SET LOWCASE ASIS command.

Attention Key Processing

When you press the attention key while XPEDITER or your application is executing, the message `ENTER ATTENTION OPTION OR HELP FOR LIST OF OPTIONS` is displayed. If you enter HELP, a screen containing information similar to Figure 2-2 on page 2-4 is displayed. Enter the option you want to perform.

```
ENTER:
        PAUSE    TO DYNAMICALLY INVOKE THE PAUSE COMMAND
        EXIT     TO TERMINATE THE TEST SESSION
        LOG      TO DISPLAY LOG PRIOR TO TERMINATION
        WHERE    TO DISPLAY CURRENT MODULE AND OFFSET
        GO       TO RESUME TEST EXECUTION
                 OR PRESS ENTER TO RESUME TEST EXECUTION
```

Figure 2-2. Attention Key Processing Options

Note: The options listed on the screen will be different for MPP programs.

In XPEDITER/IMS, when you enter **EXIT**, you see a system 33E abend reported as you return to the starting test panel. Message XPD1202 is recorded in the log showing the same S33E abend, which is caused by XPEDITER/IMS when it detaches the user's IMS region.

CAUTION

In XPEDITER/IMS, do not press the attention key twice. When you do this, the session is terminated with the TSO default, and you are returned either to ISPF or native TSO. In addition, XPEDITER/IMS is not terminated normally; i.e., your datasets cannot be closed safely.

When you press the attention key while XPEDITER is waiting for input (e.g., stopped at a breakpoint), the message `ATTENTION IGNORED - WAITING FOR TERMINAL INPUT` is displayed. If you press the attention key again, the message `ENTER ATTENTION TO TERMINATE OR ENTER TO CONTINUE` is displayed. Pressing attention again will take you to the READY prompt.

Chapter 3. Quick Sample Debugging Session

This chapter demonstrates some of the basic interactive debugging features of XPEDITER/TSO, using the sample program TRIMAIN, which calls TRITST and TRIRPT. This quick overview shows you how to do the following:

- Prepare the programs
- Start an interactive debugging session
- Set breakpoints
- Display file information
- Display data values
- Debug subroutines
- Analyze data flow
- Trace logic flow
- Monitor and review the execution path.

Preparing the Programs

The source for TRIMAIN, TRITST, TRIRPT, and TRIDATA (the input dataset containing the data) is provided on the XPEDITER/TSO distribution tape. The common load module, DDIO dataset, test data, and file list libraries should have already been created and/or verified by your site installer. Contact your installer for the names of these libraries and datasets.

If these datasets and libraries were not created, you must compile and link-edit the programs using the Compuware Shared Services (CSS) COBOL language processor. You must also specify the appropriate load module and DDIO dataset on the Setup screens.

Option **1** on the XPEDITER/TSO Primary Menu can be used to compile and link-edit the programs. Refer to Chapter 4, “Preparing Your Programs” on page 4-3 if you need assistance.

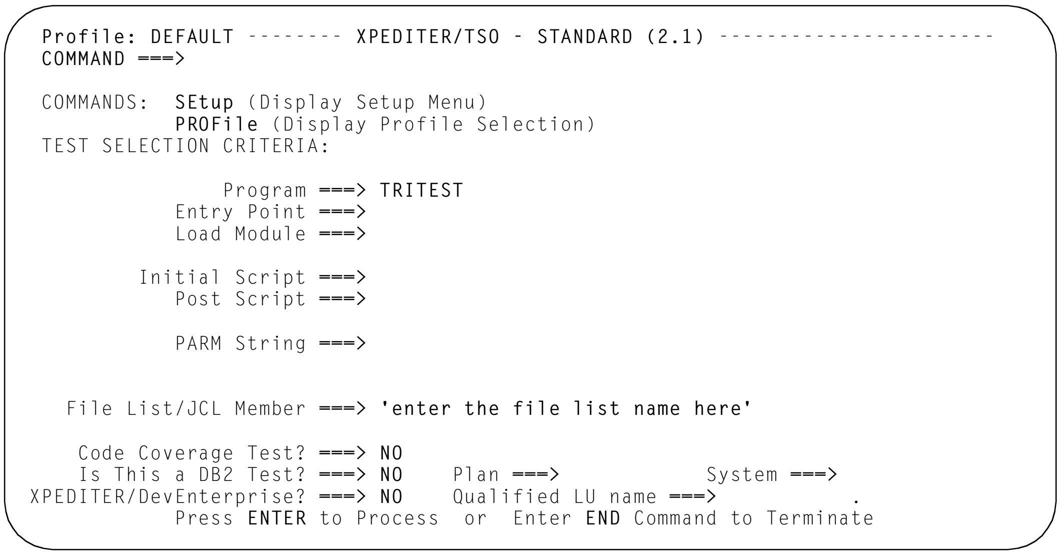
When you are ready to start the session, do the steps listed below in “Starting the Debugging Session”.

Starting the Debugging Session

1. Type **TRIMAIN** in the Profile field on the XPEDITER/TSO Primary Menu. XPEDITER creates a profile for the session, displays the Profile screen, and prompts you to enter a description for the new profile. Type **XPEDITER Sample Program TRIMAIN** in the description area and press the **End** key. The Primary Menu is redisplayed.
2. Type **2** (TSO) in the command line of the Primary Menu. The Environments Menu is displayed if this is your first time invoking an XPEDITER/TSO debugging session. Otherwise, the last environment test screen you used is displayed. To access the Environments Menu, type **SETUP** in the command line of the displayed environment test

screen. Then type option **0** on the Test Setup Menu to display the Environments Menu.

3. Type **1** (STANDARD) on the Environments Menu.
4. Type **SETUP** on the Standard test screen.
5. Type **1** (LOADLIBS) on the Setup Menu.
6. On the Load Module Libraries screen, enter the name of the application load library that contains the TRIMAIN load module.
7. Press **Enter**.
8. Type **2** (DDIO) on the Setup Menu.
9. On the DDIO Files screen, enter the DDIO dataset name that contains the source listing member of TRIMAIN.
10. Press **Enter**.
11. Type **END** or press **PF3** to return to the Standard environment test screen.
12. Specify the name of the program and the name of the file list or the JCL containing the names of the files required by your program. Complete the screen as shown in Figure 3-1.



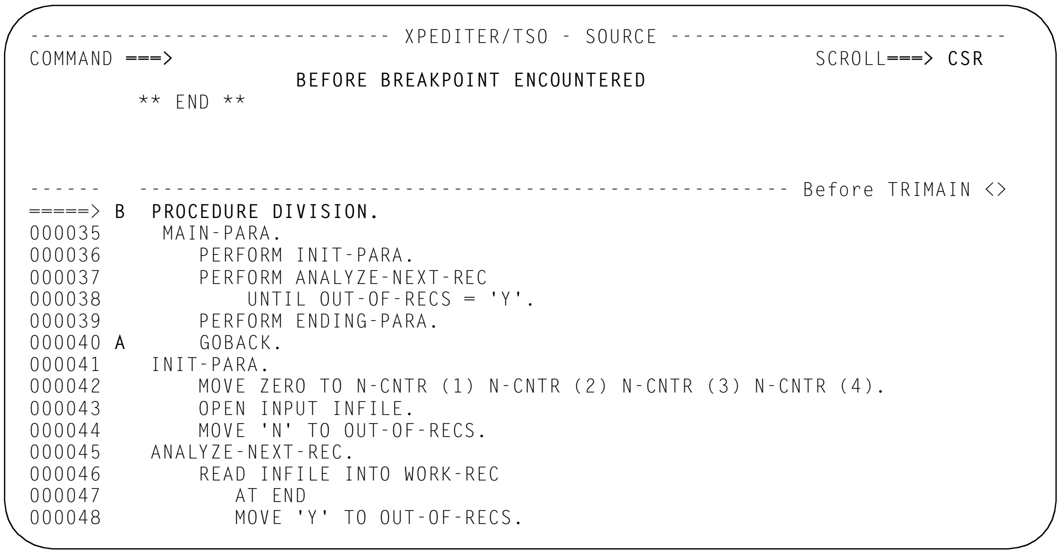
```
 Profile: DEFAULT -------- XPEDITER/TSO - STANDARD (2.1) -----------------------
 COMMAND ===>

 COMMANDS:  SEtup (Display Setup Menu)
            PROFile (Display Profile Selection)
 TEST SELECTION CRITERIA:

                Program ===> TRITEST
            Entry Point ===>
            Load Module ===>

         Initial Script ===>
            Post Script ===>

            PARM String ===>


   File List/JCL Member ===> 'enter the file list name here'

    Code Coverage Test? ===> NO
    Is This a DB2 Test? ===> NO    Plan ===>            System ===>
XPEDITER/DevEnterprise? ===> NO    Qualified LU name ===>          .
            Press ENTER to Process  or  Enter END Command to Terminate
```

Figure 3-1. Standard Test Screen

13. Press **Enter** to begin the XPEDITER/TSO debugging session. The message area contains the lines `Allocating XPEDITER/TSO Datasets`, then `Allocating User Datasets`. On a blank screen, the message `Entering XPEDITER/TSO Test Environment` appears.

XPEDITER/TSO processes the file list, allocates the necessary datasets, loads the program TRIMAIN, and displays the COBOL source listing. You should be able to see the message `Before Breakpoint Encountered` with the execution arrow (=====>) pointing to the PROCEDURE DIVISION statement. This means that TRIMAIN stopped before beginning execution of the program. Also, a left/right scroll indicator (e.g., -- Before TRIMAIN <>) appears on the execution status line. A double arrow indicates that scrolling is allowed both left and right. An example of the source display is shown in Figure 3-2 on page 3-3.

Note: XPEDITER/TSO automatically sets a before breakpoint (B) at the entry to the program and an after breakpoint (A) at the exit to the program.



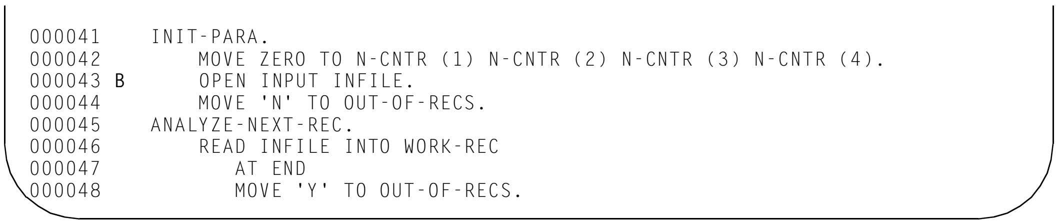
```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                     SCROLL===> CSR
                      BEFORE BREAKPOINT ENCOUNTERED
         ** END **



------   ------------------------------------------------------ Before TRIMAIN <>
=====> B   PROCEDURE DIVISION.
000035      MAIN-PARA.
000036         PERFORM INIT-PARA.
000037         PERFORM ANALYZE-NEXT-REC
000038             UNTIL OUT-OF-RECS = 'Y'.
000039         PERFORM ENDING-PARA.
000040 A       GOBACK.
000041     INIT-PARA.
000042         MOVE ZERO TO N-CNTR (1) N-CNTR (2) N-CNTR (3) N-CNTR (4).
000043         OPEN INPUT INFILE.
000044         MOVE 'N' TO OUT-OF-RECS.
000045     ANALYZE-NEXT-REC.
000046         READ INFILE INTO WORK-REC
000047            AT END
000048            MOVE 'Y' TO OUT-OF-RECS.
```

Figure 3-2. Source Screen Showing TRIMAIN Program

Setting Breakpoints

You can control program execution by using XPEDITER/TSO commands that set breakpoints. A breakpoint is a certain location in your program where you want program execution to stop.

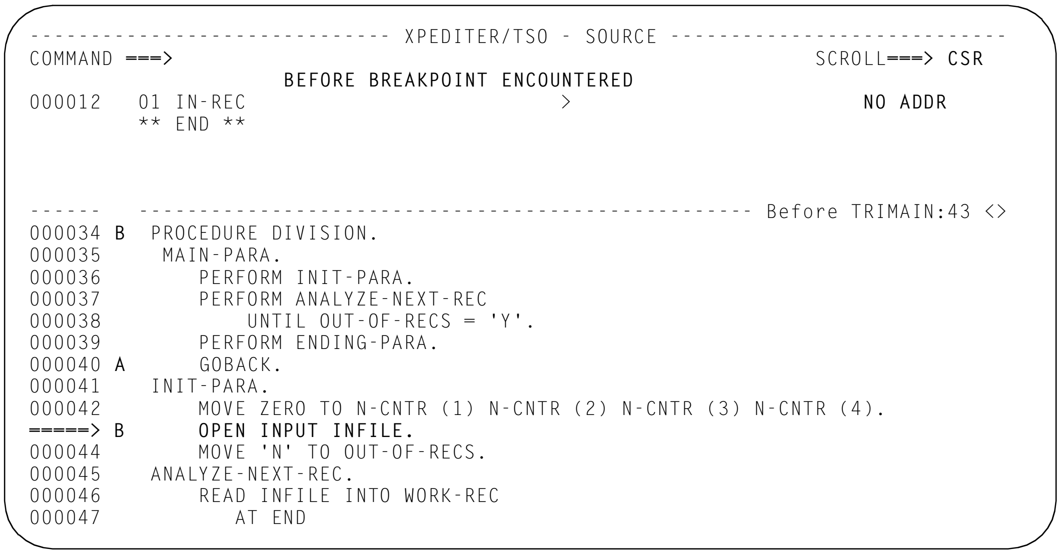
A simple way to enter a breakpoint command is to type it in the line command area. Move the cursor to the compiler-generated statement number 43 at the OPEN verb, then type over the statement number with the **B** (Before) line command and press **Enter**. As shown in Figure 3-3, a B appears in column 9 on statement 43, indicating that a before breakpoint has been set. This breakpoint causes program execution to pause before executing the OPEN statement.



```
000041     INIT-PARA.
000042         MOVE ZERO TO N-CNTR (1) N-CNTR (2) N-CNTR (3) N-CNTR (4).
000043 B       OPEN INPUT INFILE.
000044         MOVE 'N' TO OUT-OF-RECS.
000045     ANALYZE-NEXT-REC.
000046         READ INFILE INTO WORK-REC
000047            AT END
000048            MOVE 'Y' TO OUT-OF-RECS.
```

Figure 3-3. Entering a Before Breakpoint at Statement 43

Press **PF12** or type **GO** to execute TRIMAIN until the breakpoint is reached.The program stops at statement 43, where the execution arrow is pointing and IN-REC is automatically kept in the Keep window. The execution status field on the fourth line also shows that execution is paused Before TRIMAIN:43, as shown in Figure 3-4 on page 3-4.



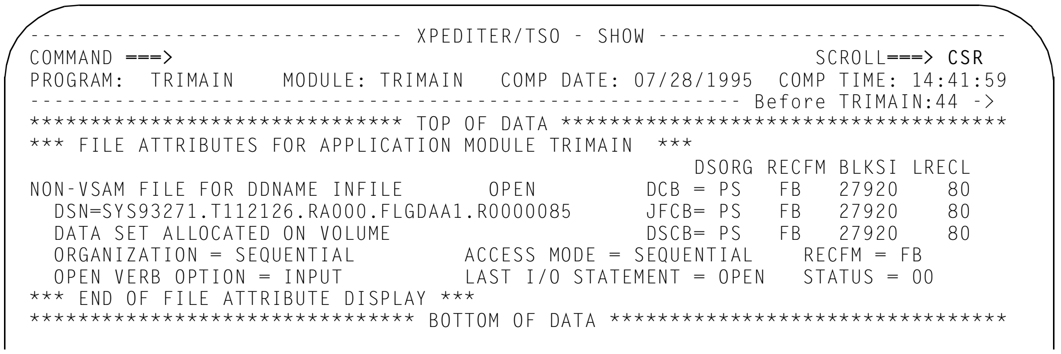
```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                    SCROLL===> CSR
                     BEFORE BREAKPOINT ENCOUNTERED
000012   01 IN-REC                         >                        NO ADDR
         ** END **



------   --------------------------------------------------- Before TRIMAIN:43 <>
000034 B  PROCEDURE DIVISION.
000035     MAIN-PARA.
000036        PERFORM INIT-PARA.
000037        PERFORM ANALYZE-NEXT-REC
000038            UNTIL OUT-OF-RECS = 'Y'.
000039        PERFORM ENDING-PARA.
000040 A      GOBACK.
000041    INIT-PARA.
000042        MOVE ZERO TO N-CNTR (1) N-CNTR (2) N-CNTR (3) N-CNTR (4).
=====> B      OPEN INPUT INFILE.
000044        MOVE 'N' TO OUT-OF-RECS.
000045    ANALYZE-NEXT-REC.
000046        READ INFILE INTO WORK-REC
000047           AT END
```

Figure 3-4. Program Stopped at Before Breakpoint on Statement 43

Press **PF9** (GO 1) to execute the next statement.The program stops at statement 44 after the input file is opened and before the OUT-OF-RECS switch is set to N. Since the execution arrow is now paused on statement 44, IN-REC disappears from the Keep window and OUT-OF-RECS is automatically displayed.

Displaying File Information

Type the SHOW FILE command in the primary command line.The ddnames and DCB parameters that are specified in the JCL statements are listed, together with the file I/O status, as shown in Figure 3-5.



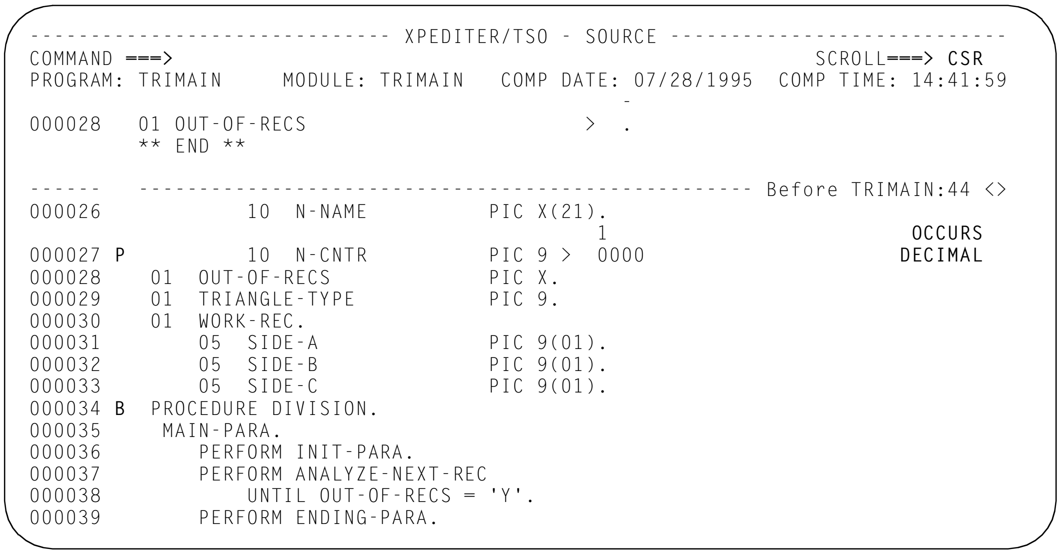
```
------------------------------- XPEDITER/TSO - SHOW ----------------------------
COMMAND ===>                                                    SCROLL===> CSR
PROGRAM:  TRIMAIN    MODULE: TRIMAIN   COMP DATE: 07/28/1995  COMP TIME: 14:41:59
-------------------------------------------------------------- Before TRIMAIN:44 ->
******************************* TOP OF DATA ***********************************
*** FILE ATTRIBUTES FOR APPLICATION MODULE TRIMAIN  ***
                                                     DSORG RECFM BLKSI LRECL
NON-VSAM FILE FOR DDNAME INFILE       OPEN          DCB = PS    FB   27920    80
  DSN=SYS93271.T112126.RA000.FLGDAA1.R0000085      JFCB= PS    FB   27920    80
  DATA SET ALLOCATED ON VOLUME                     DSCB= PS    FB   27920    80
  ORGANIZATION = SEQUENTIAL         ACCESS MODE = SEQUENTIAL    RECFM = FB
  OPEN VERB OPTION = INPUT          LAST I/O STATEMENT = OPEN   STATUS = 00
*** END OF FILE ATTRIBUTE DISPLAY ***
******************************** BOTTOM OF DATA ********************************
```

Figure 3-5. File Attributes Displayed by the SHOW FILE Command

Press **PF3** (END) to return to the Source screen.

Displaying Data Values

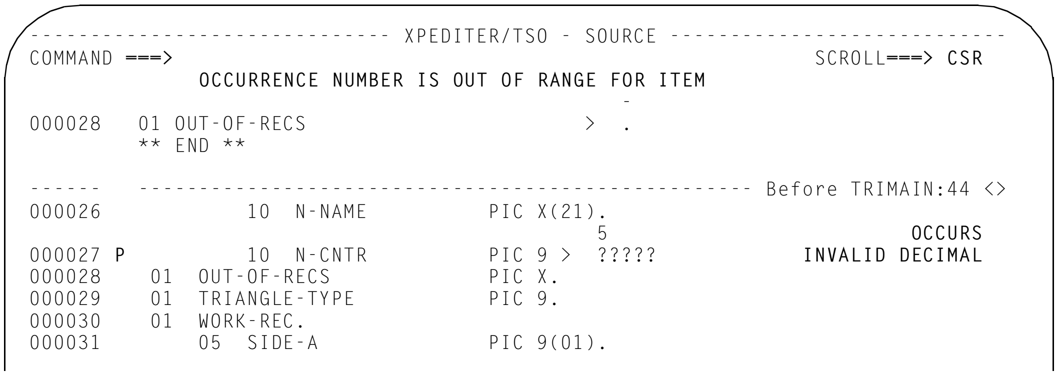
Move the cursor to statement 42 where the table (N-CNTR) is initialized with zeros. Type over the statement number with the **P** (Peek) line command, and press **Enter**. The screen automatically scrolls to the DATA DIVISION statement where the table is defined, inserts a P in column 9, and displays the occurrence and value of N-CNTR, as shown in Figure 3-6 on page 3-5.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                      SCROLL===> CSR
PROGRAM: TRIMAIN     MODULE: TRIMAIN   COMP DATE: 07/28/1995  COMP TIME: 14:41:59
                                                         -
000028   01 OUT-OF-RECS                      >   .
         ** END **

------   -------------------------------------------------- Before TRIMAIN:44 <>
000026            10  N-NAME           PIC X(21).
                                                1                         OCCURS
000027 P          10  N-CNTR           PIC 9 >  0000                     DECIMAL
000028    01  OUT-OF-RECS              PIC X.
000029    01  TRIANGLE-TYPE            PIC 9.
000030    01  WORK-REC.
000031        05  SIDE-A               PIC 9(01).
000032        05  SIDE-B               PIC 9(01).
000033        05  SIDE-C               PIC 9(01).
000034 B  PROCEDURE DIVISION.
000035     MAIN-PARA.
000036        PERFORM INIT-PARA.
000037        PERFORM ANALYZE-NEXT-REC
000038            UNTIL OUT-OF-RECS = 'Y'.
000039        PERFORM ENDING-PARA.
```

Figure 3-6. Displaying the Data Content of N-CNTR

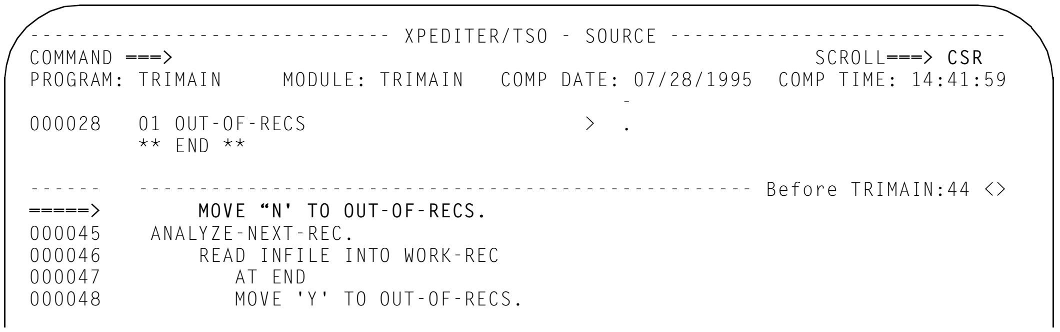
Tab to the occurrence number and type over the 1 with a **2** and press **Enter**.Continue typing over the occurrence number, each time adding 1 to the previous number. Eventually, XPEDITER/TSO displays the warning shown in Figure 3-7, indicating that the index boundary has been reached.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                      SCROLL===> CSR
              OCCURRENCE NUMBER IS OUT OF RANGE FOR ITEM
                                                         -
000028   01 OUT-OF-RECS                      >   .
         ** END **

------   -------------------------------------------------- Before TRIMAIN:44 <>
000026            10  N-NAME           PIC X(21).
                                                5                         OCCURS
000027 P          10  N-CNTR           PIC 9 >  ?????            INVALID DECIMAL
000028    01  OUT-OF-RECS              PIC X.
000029    01  TRIANGLE-TYPE            PIC 9.
000030    01  WORK-REC.
000031        05  SIDE-A               PIC 9(01).
```

Figure 3-7. Message Indicating That the Index Boundary Has Been Reached

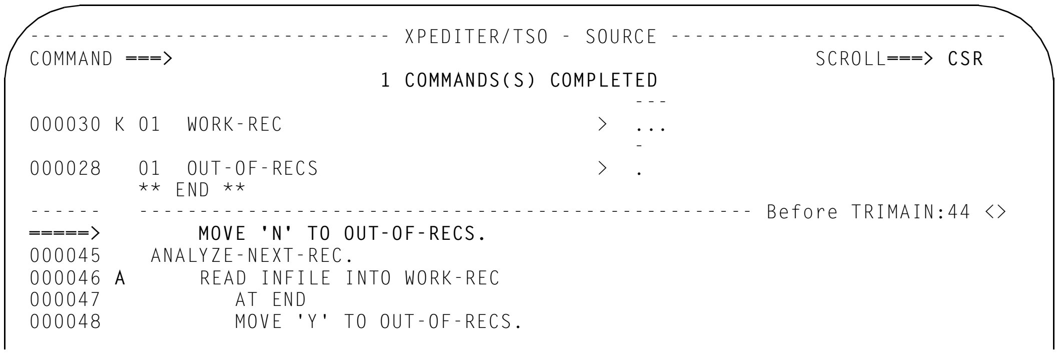
Press **PF6** (LOCATE *****) to scroll to the current execution arrow. Refer to Figure 3-8.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                      SCROLL===> CSR
PROGRAM: TRIMAIN     MODULE: TRIMAIN   COMP DATE: 07/28/1995  COMP TIME: 14:41:59
                                                         -
000028   01 OUT-OF-RECS                      >   .
         ** END **

------   -------------------------------------------------- Before TRIMAIN:44 <>
=====>        MOVE “N' TO OUT-OF-RECS.
000045    ANALYZE-NEXT-REC.
000046        READ INFILE INTO WORK-REC
000047           AT END
000048           MOVE 'Y' TO OUT-OF-RECS.
```

Figure 3-8. TRIMAIN After Entering the LOCATE * Command

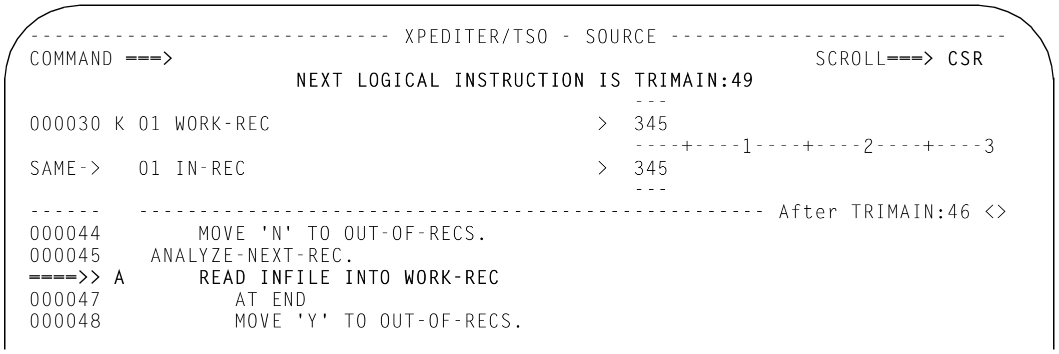
Enter an **A** line command on statement 46, setting an after breakpoint at the READ statement.Next, type the **K2** line command on statement 46 to display the contents of working storage for WORK-REC (the second variable identified on line 46). Press **Enter**. The display is shown in Figure 3-9 on page 3-6. The K in column 9 of the window indicates that it is an explicitly kept item and distinguishes it from the automatically kept data.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                        SCROLL===> CSR
                           1 COMMANDS(S) COMPLETED
                                                       ---
000030 K 01  WORK-REC                              >   ...
                                                       -
000028   01  OUT-OF-RECS                           >   .
         ** END **
------   -------------------------------------------------- Before TRIMAIN:44 <>
=====>        MOVE 'N' TO OUT-OF-RECS.
000045     ANALYZE-NEXT-REC.
000046 A       READ INFILE INTO WORK-REC
000047            AT END
000048            MOVE 'Y' TO OUT-OF-RECS.
```

Figure 3-9. Adding an Explicit Keep (WORK-REC) to the Keep Window

Press **PF12** (GO) to execute TRIMAIN. As shown in Figure 3-10, you can see that record 345 was read when the READ verb was executed. Note that the automatic keep of WORK-REC is only partially displayed since the window contains only 5 lines. To scroll the window, move the cursor into the window and use the PF7 and PF8 keys.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                        SCROLL===> CSR
                    NEXT LOGICAL INSTRUCTION IS TRIMAIN:49
                                                       ---
000030 K 01 WORK-REC                               >   345
                                                       ----+----1----+----2----+----3
SAME->   01 IN-REC                                 >   345
                                                       ---
------   --------------------------------------------------- After TRIMAIN:46 <>
000044        MOVE 'N' TO OUT-OF-RECS.
000045     ANALYZE-NEXT-REC.
====>> A       READ INFILE INTO WORK-REC
000047            AT END
000048            MOVE 'Y' TO OUT-OF-RECS.
```

Figure 3-10. Displaying a Variable in a Keep Window

Press **PF12** again. Paragraph ANALYZE-NEXT-REC is performed until EOF, and the next record, 789, is read in the second time through the loop. WORK-REC is updated to reflect the change.As you execute your program, XPEDITER/TSO updates the Keep window to reflect the current values of the explicitly kept data.

Debugging Subroutines

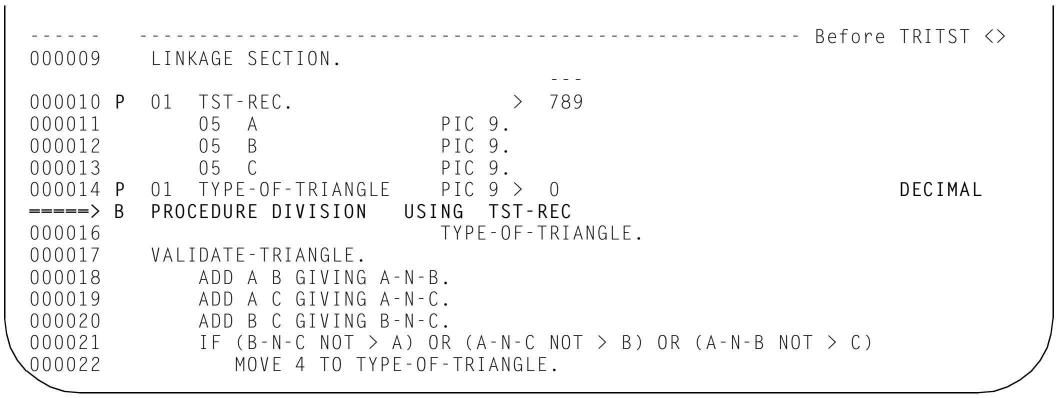
Statement 51 shows that the program TRIMAIN calls TRITST and passes parameters WORK-REC and TRIANGLE-TYPE. In order to examine how TRITST is processing these parameters, you can set a breakpoint at the beginning of TRITST to gain control of the execution. Type the following command on the primary command line:

```
BEFORE TRITST:
```

The colon (:) after the program name indicates program qualification. Press **PF12** (GO) to execute the program. XPEDITER/TSO sets a before module breakpoint at the beginning of the TRITST program and pauses execution at the PROCEDURE DIVISION USING statement in TRITST. Now, type the following command on the primary command line:

```
PEEK LINKAGE
```

The Linkage Section shows that the correct values, 789 for TST-REC and 0 for TYPE-OF-TRIANGLE, were passed from the driver TRIMAIN.



```
------   ------------------------------------------------------- Before TRITST <>
000009    LINKAGE SECTION.
                                          ---
000010 P  01  TST-REC.                 >  789
000011        05  A             PIC 9.
000012        05  B             PIC 9.
000013        05  C             PIC 9.
000014 P  01  TYPE-OF-TRIANGLE  PIC 9 >  0                          DECIMAL
=====> B  PROCEDURE DIVISION   USING  TST-REC
000016                          TYPE-OF-TRIANGLE.
000017    VALIDATE-TRIANGLE.
000018        ADD A B GIVING A-N-B.
000019        ADD A C GIVING A-N-C.
000020        ADD B C GIVING B-N-C.
000021        IF (B-N-C NOT > A) OR (A-N-C NOT > B) OR (A-N-B NOT > C)
000022           MOVE 4 TO TYPE-OF-TRIANGLE.
```

Figure 3-11. Displaying the Linkage Section in the Called Module TRITST

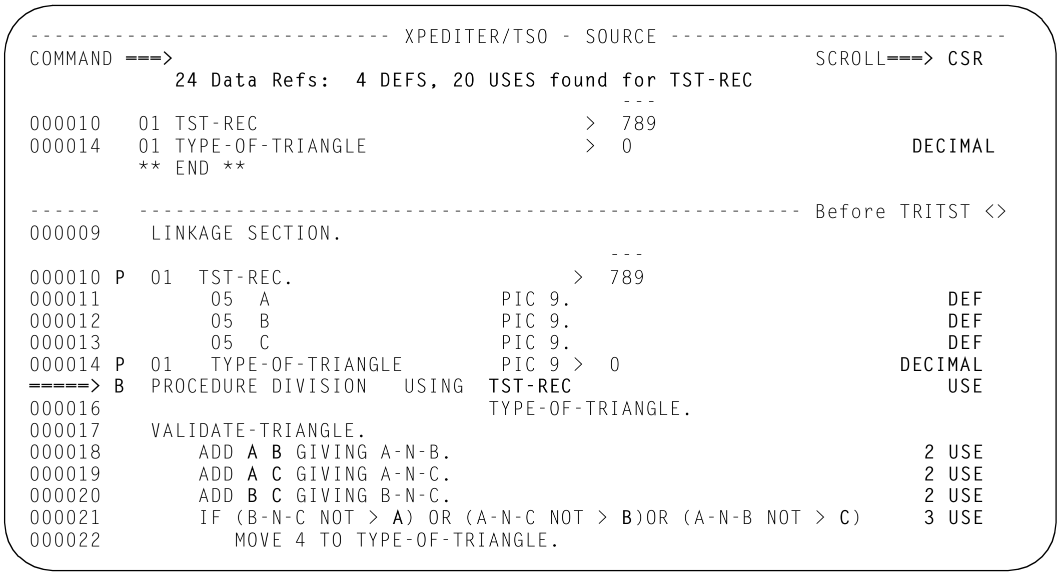
Analyzing Data Flow

To better understand how the parameters are processed in the subroutine, XPEDITER/TSO allows you to cross-reference data and to analyze the data flow in your program. The 01 level for TST-REC has three 05 levels: A, B, and C. Essentially, the elementary items are the aliases of a group item. Type the following command on the primary command line:

```
FIND TST-REC ALIAS ALL
```

XPEDITER/TSO highlights all the statements that reference (DEFINE, MODIFY, USE) TST-REC and its aliases. The message shown in Figure 3-12 is issued, which states how many data definitions were found.

Note: Enhanced FIND cannot be used in nested programs. Only a string FIND is valid.



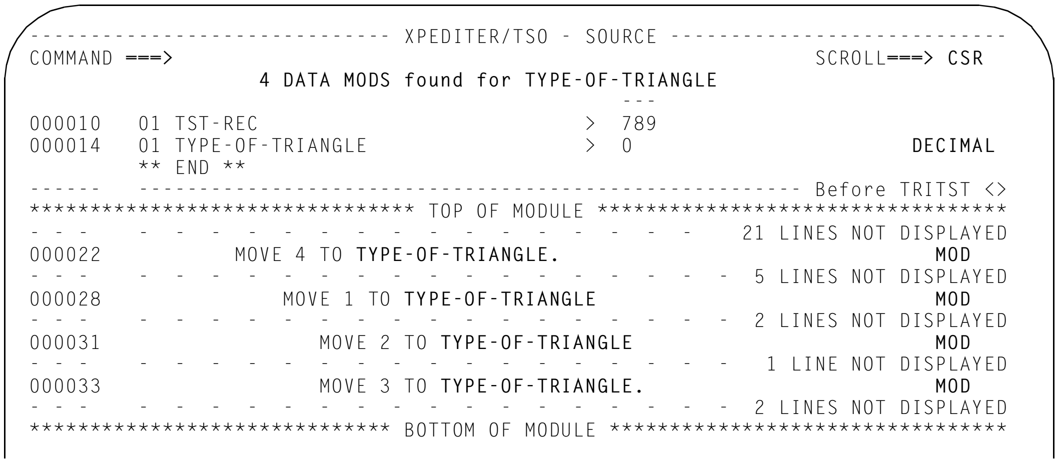
```
----------------------------- XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                  SCROLL===> CSR
            24 Data Refs:  4 DEFS, 20 USES found for TST-REC
                                              ---
000010   01 TST-REC                       >  789
000014   01 TYPE-OF-TRIANGLE              >  0                        DECIMAL
         ** END **

------   ------------------------------------------------------- Before TRITST <>
000009    LINKAGE SECTION.
                                               ---
000010 P  01  TST-REC.                      >  789
000011         05  A                 PIC 9.                             DEF
000012         05  B                 PIC 9.                             DEF
000013         05  C                 PIC 9.                             DEF
000014 P  01   TYPE-OF-TRIANGLE      PIC 9 >  0                      DECIMAL
=====> B  PROCEDURE DIVISION   USING  TST-REC                           USE
000016                              TYPE-OF-TRIANGLE.
000017    VALIDATE-TRIANGLE.
000018        ADD A B GIVING A-N-B.                                   2 USE
000019        ADD A C GIVING A-N-C.                                   2 USE
000020        ADD B C GIVING B-N-C.                                   2 USE
000021        IF (B-N-C NOT > A) OR (A-N-C NOT > B)OR (A-N-B NOT > C)     3 USE
000022           MOVE 4 TO TYPE-OF-TRIANGLE.
```

Figure 3-12. Finding Statements That Reference TST-REC

The analysis concludes that parameter TST-REC is used, but never modified in the subroutine. What about parameter TYPE-OF-TRIANGLE? Type the following command on the primary command line:

```
FIND TYPE-OF-TRIANGLE MOD ALL EXCLUDE
```

The EXCLUDE keyword was used to exclude lines that do not meet the search criteria. As shown in Figure 3-13, the message `4 DATA MODS found for TYPE-OF-TRIANGLE` is displayed in the message line. This indicates the parameter was modified four times.



```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                      SCROLL===> CSR
                   4 DATA MODS found for TYPE-OF-TRIANGLE
                                                   ---
000010   01 TST-REC                           >  789
000014   01 TYPE-OF-TRIANGLE                  >  0                       DECIMAL
         ** END **
------   ------------------------------------------------------ Before TRITST <>
******************************** TOP OF MODULE *********************************
- - -    - - - - - - - - - - - - - - - - - - - - - -      21 LINES NOT DISPLAYED
000022          MOVE 4 TO TYPE-OF-TRIANGLE.                                  MOD
- - -    - - - - - - - - - - - - - - - - - - - - - - -     5 LINES NOT DISPLAYED
000028              MOVE 1 TO TYPE-OF-TRIANGLE                               MOD
- - -    - - - - - - - - - - - - - - - - - - - - - - -     2 LINES NOT DISPLAYED
000031                 MOVE 2 TO TYPE-OF-TRIANGLE                            MOD
- - -    - - - - - - - - - - - - - - - - - - - - - - -      1 LINE NOT DISPLAYED
000033                 MOVE 3 TO TYPE-OF-TRIANGLE.                           MOD
- - -    - - - - - - - - - - - - - - - - - - - - - - -     2 LINES NOT DISPLAYED
****************************** BOTTOM OF MODULE ********************************
```

Figure 3-13. Result of FIND TYPE-OF-TRIANGLE MOD EXCLUDE Command

The FIND command under XPEDITER/TSO is sensitive to COBOL-structure keywords as well as data reference keywords. For instance, you can use the FIND command to search conditional statements or I/O statements. The highlighting effect helps you capture the program logic and understand what the program does.

To reset all excluded lines in the program, enter **END** or press **PF3**.

Tracing Logic Flow

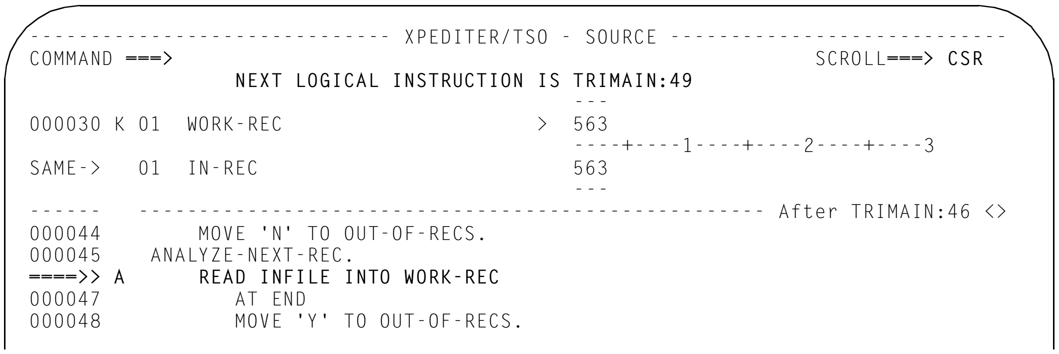
Subroutine TRITST evaluates the type of triangle by using TST-REC and then it updates TYPE-OF-TRIANGLE. Paragraph DETERMINE-TYPE (statement 23) has a nested IF structure. XPEDITER/TSO can automatically trace the logic flow to show which path was chosen. Type the following command in the primary command line to control the tracing speed:

```
SET DELAY 1
```

Then type the following command on the primary command line and press **Enter**.

```
GO TRACE
```

Tracing pauses when the after breakpoint in TRIMAIN is reached, as shown in Figure 3-14.



```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                      SCROLL===> CSR
                 NEXT LOGICAL INSTRUCTION IS TRIMAIN:49
                                               ---
000030 K 01  WORK-REC                     >  563
                                             ----+----1----+----2----+----3
SAME->   01  IN-REC                          563
                                             ---
------   --------------------------------------------------- After TRIMAIN:46 <>
000044        MOVE 'N' TO OUT-OF-RECS.
000045    ANALYZE-NEXT-REC.
====>> A      READ INFILE INTO WORK-REC
000047           AT END
000048           MOVE 'Y' TO OUT-OF-RECS.
```

Figure 3-14. Tracing is Paused for After Breakpoint in Calling Module TRIMAIN

Monitoring and Reviewing the Execution Path

Type the following command on the primary command line to activate review mode for all COBOL modules with a source listing member:

```
MONITOR ALL
```

Press **PF12** (GO) to start execution, followed by another **PF12** to continue execution.When the after breakpoint is reached in TRIMAIN, type the following command on the primary command line to change the direction of execution processing:

```
REVERSE
```

The execution status line shows that XPEDITER/TSO is reviewing in the reverse direction. See Figure 3-15.

```
 ------   -------------------------------------------Reverse -
After TRIMAIN:46 <>
 000044        MOVE 'N' TO OUT-OF-RECS.
 000045    ANALYZE-NEXT-REC.
 ====>> A      READ INFILE INTO WORK-REC
 000047           AT END
```

Figure 3-15. Review Mode Execution

Now step through each statement backwards by pressing **PF9** (GO 1) several times.Data values in the Keep window redisplay the original state as the MOVE, ADD, and READ verbs are being "undone."

You can remove explicitly kept data from the Keep window by typing the following command in the primary command line:

```
DELETE KEEP
```

If you want to remove a certain data item from the Keep window, type the **D** line command on the appropriate line.Press **PF4** (EXIT) to exit the debugging session and to return to the Standard test screen.

Chapter 4. Getting Started

This chapter discusses the following:

- The XPEDITER/TSO Primary Menu.
- Preparing your programs to be processed with XPEDITER/TSO and XPEDITER/IMS. Refer to “Preparing Your Programs” on page 4-3.
- Starting an XPEDITER debugging session in interactive mode. Refer to “Starting an Interactive Session” on page 4-12.
- Starting an XPEDITER debugging session through the batch connect facility (for interactive or unattended batch debugging). Refer to “Starting a Batch Connect Session” on page 4-17.
- Manually changing your execution JCL to debug with XPEDITER/TSO in interactive (batch connect) or unattended batch mode. Refer to “Starting a Session With Batch JCL” on page 4-37.

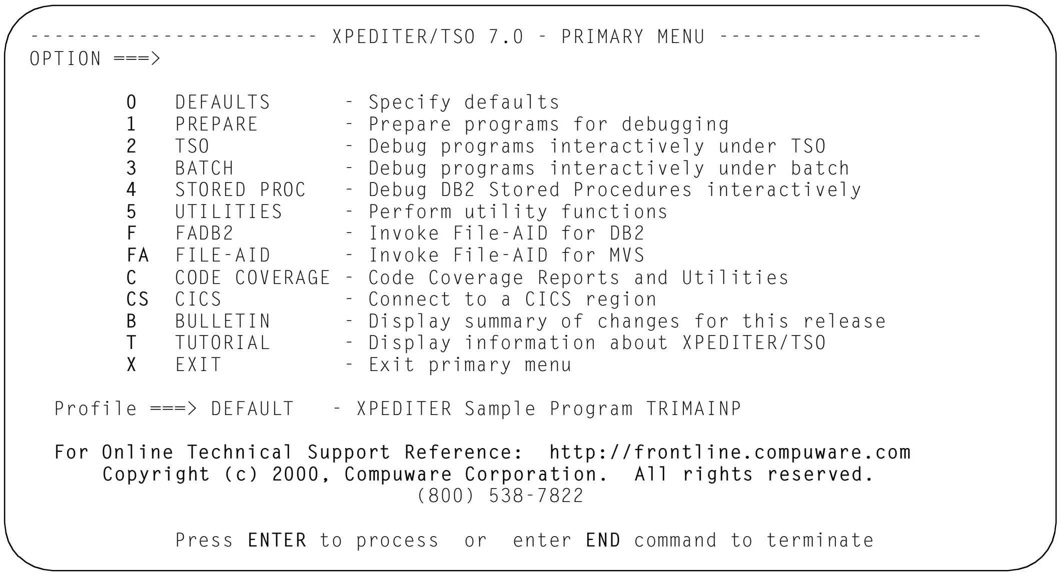
  Note: XPEDITER/IMS can only be invoked in interactive mode.
- Using script files for test session management. Refer to “Test Session Management Using Scripts” on page 4-42.
- Accessing other systems from XPEDITER. Refer to “Accessing Other Systems From XPEDITER/TSO” on page 4-48.

The XPEDITER/TSO Primary Menu

The Primary Menu is the first screen displayed upon entry to XPEDITER/TSO. The options on the Primary Menu are used to prepare the programs for execution, provide all the information needed to execute the program, and invoke the test session. It also provides a gateway to other Compuware products. The Primary Menu for XPEDITER is shown in Figure 4-1 on page 4-2.

Notes:

1. Only the options that have been installed at your site appear on your Primary Menu.
2. First time users will automatically see the bulletin containing the new features for this release.



```
----------------------- XPEDITER/TSO 7.0 - PRIMARY MENU ---------------------
OPTION ===>

        0   DEFAULTS      - Specify defaults
        1   PREPARE       - Prepare programs for debugging
        2   TSO           - Debug programs interactively under TSO
        3   BATCH         - Debug programs interactively under batch
        4   STORED PROC   - Debug DB2 Stored Procedures interactively
        5   UTILITIES     - Perform utility functions
        F   FADB2         - Invoke File-AID for DB2
        FA  FILE-AID      - Invoke File-AID for MVS
        C   CODE COVERAGE - Code Coverage Reports and Utilities
        CS  CICS          - Connect to a CICS region
        B   BULLETIN      - Display summary of changes for this release
        T   TUTORIAL      - Display information about XPEDITER/TSO
        X   EXIT          - Exit primary menu

  Profile ===> DEFAULT   - XPEDITER Sample Program TRIMAINP

  For Online Technical Support Reference:  http://frontline.compuware.com
      Copyright (c) 2000, Compuware Corporation.  All rights reserved.
                              (800) 538-7822

              Press ENTER to process  or  enter END command to terminate
```

Figure 4-1. Primary Menu for XPEDITER

The Primary Menu has the following options:

0 DEFAULTS
Specifies defaults for your terminal, PF keys, profile, and screen colors. Refer to Appendix D, "Specifying Session Defaults."

1 PREPARE
Accesses the Program Preparation Menu where you can:

- Convert your JCL to compile your programs with the Compuware Shared Services COBOL language processor.
- DB2 precompile, compile, link-edit, and EXEC CICS translation.
- Bind application plans with File-AID for DB2.
- Create and edit lists of the files and databases that have to be allocated for your program to execute properly.

2 TSO
Invokes an interactive XPEDITER/TSO and XPEDITER/IMS debugging session. Refer to “Starting an Interactive Session” on page 4-12.

3 BATCH
Invokes an XPEDITER/TSO test session in batch and allows you to interactively connect to the batch job for debugging. Refer to “Starting a Batch Connect Session” on page 4-17.

4 STORED PROC
Invokes a DB2 Stored Procedure debugging session.

5 UTILITIES
Accesses miscellaneous utility functions that can be used to query available region size, list dataset allocations, maintain DDIO datasets, convert script files (INCLUDES) and convert profiles from previous releases. Refer to Appendix E, "XPEDITER/TSO Utilities" on page E-1.

F FADB2
Invokes File-AID for DB2, which is Compuware’s powerful and easy to use DB2 database management, application development, and performance analysis tool. Refer to the *File-AID for DB2 Reference Manual* for information about using File-AID for DB2.

FA FILE-AID
Invokes File_AID for MVS. Refer to the *File-AID for MVS Reference Manual* for information about using File-AID for MVS.

C CODE COVERAGE
Invokes XPEDITER/Code Coverage, a product which is only available if you have purchased it. When you run a test using Code Coverage, you are able to track every statement you have executed in the normal work flow while running the test. Refer to the *XPEDITER/Code Coverage Mainframe User/Reference Guide* for more information about using XPEDITER/Code Coverage.

CS CICS
Connects to a CICS region for testing or any other CICS function. Refer to "Accessing Other Systems From XPEDITER/TSO" on page 4-48.

B BULLETIN
Displays a summary of changes for this release. This option is automatically displayed when you first log onto the system. A similar version of this information is contained in the Preface of this manual.

T TUTORIAL
Provides general or specific information about an XPEDITER/TSO command, screen, or message.

Note: Help can also be accessed by pressing **PF1** or entering **HELP** on the command line of any screen.

Once you are in the Tutorial, you can continue to press **Enter** to see subsequent pages of help text. The UP, BACK, NEXT, INDEX, and HELP subcommands can be used to quickly locate a particular topic.

Use the END command to terminate the HELP function.

X EXIT
Exits XPEDITER/TSO.

PROFile
Specifies the current profile. Refer to "Using Profiles" on page 4-13.

Preparing Your Programs

XPEDITER/TSO provides online program preparation facilities that can be used to precompile, compile, and link-edit your programs, convert your existing JCL to compile your programs with Compuware's Shared Services (CSS) COBOL language processor, bind programs for DB2 processing, and prepare file allocation lists.

To debug a program symbolically at the source level with XPEDITER, it must be compiled with the COBOL language processor. When activated, the language processor (LP) accepts the source code, invokes the specified compiler, and waits until the compile step is complete. It then analyzes the program listing, produces a source listing member, and writes the source listing member to the DDIO file. XPEDITER refers to the symbolic information and maps the source representation to the load module, so that you can debug your program interactively at the source level.

Notes:

1. XPEDITER requires specific compile options which may vary depending on the language type and actual version being used. A user should always refer to the appropriate *CSS User/Reference Guide* for information related to any required compiler options. Examples are: (1) Options NOTEST and NONUMBER are required for COBOL; (2) Option NOTEST is required for PL/I, and (3) Options ESD and LIST are required for Assembler. The *guide* is required to assist users in manually converting Compile and Link JCL to execute the CSS Language Processor. The *guide* also gives most users information on how to set up the Language Processor parameters to write a matching source listing to a DDIO file.

2. If you use a postprocessed source listing with XPEDITER/TSO, you may lose some XPEDITER/TSO functions. If the source listing contains the COPY SUPPRESS statement, the XPEDITER/TSO commands FIND, PEEK, and KEEP will not work properly.

   If the source listing contains the PRINT NOGEN statement, the XPEDITER/TSO GEN command will not work properly.

   The postprocessed source listing must be compiled with the options required for XPEDITER/TSO; otherwise, the results are unpredictable.

3. If your program is a DB2 program and XPEDITER for DB2 Extension and File-AID for DB2 are installed at your site and you want to dynamically insert SQL statements or use the EXPLAIN command, you must precompile, compile, link-edit your program, and bind your program application plan. Refer to Appendix F, "Binding The Application Plan" on page F-1 for information about binding.

To access the XPEDITER/TSO program preparation facilities, type **1** (PREPARE) on the XPEDITER/TSO Primary Menu. The Program Preparation Menu shown in Figure 4-2 is displayed.

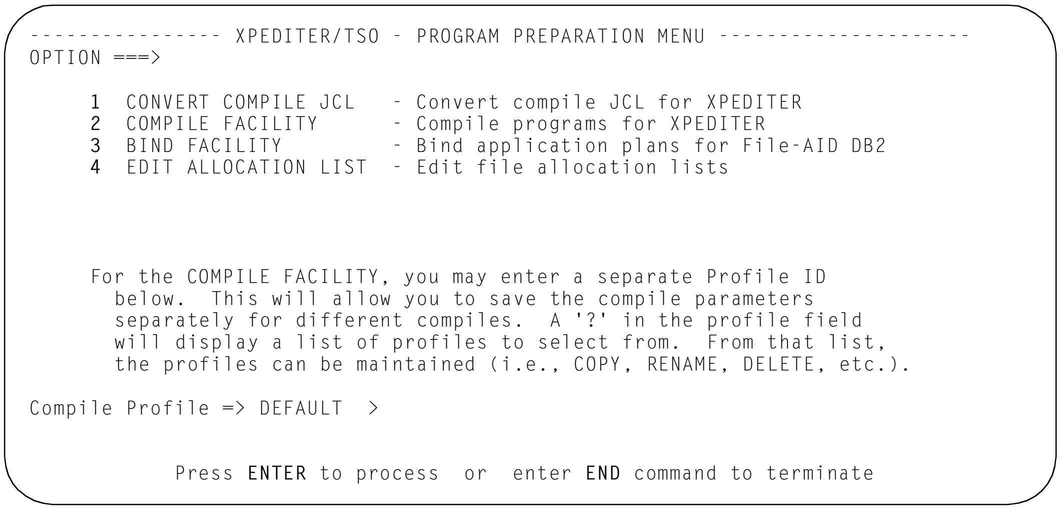

```
---------------- XPEDITER/TSO - PROGRAM PREPARATION MENU ---------------------
OPTION ===>

     1  CONVERT COMPILE JCL   - Convert compile JCL for XPEDITER
     2  COMPILE FACILITY      - Compile programs for XPEDITER
     3  BIND FACILITY         - Bind application plans for File-AID DB2
     4  EDIT ALLOCATION LIST  - Edit file allocation lists




     For the COMPILE FACILITY, you may enter a separate Profile ID
       below.  This will allow you to save the compile parameters
       separately for different compiles.  A '?' in the profile field
       will display a list of profiles to select from.  From that list,
       the profiles can be maintained (i.e., COPY, RENAME, DELETE, etc.).

Compile Profile => DEFAULT  >


            Press ENTER to process  or  enter END command to terminate
```

Figure 4-2. Program Preparation Menu

The options on this menu are:

1 CONVERT COMPILE JCL
Automatically converts your compile JCL to compile your programs with the COBOL language processor. After the JCL is converted, it can be submitted for processing.

2 COMPILE FACILITY
Precompiles DB2 statements, translates CICS commands, compiles, and link-edits your programs based on the information you enter. If you are processing in batch, the JCL is automatically built to compile with the language processor and the job is submitted. If you are processing in foreground, the compile is performed using ISPF and TSO command functions.

3 BIND FACILITY
Binds your DB2 program application plan.

If the XPEDITER for DB2 Extension and File-AID for DB2 are installed at your site and you will be using either product during the debugging session, bind the program application plan with File-AID for DB2. Refer to Appendix F, "Binding the Application Plan" for information about binding.

4 EDIT ALLOCATION LIST
Invokes the File Allocation Utility (FAU) to create and allocate the files your program will need to execute. The FAU is only used when you are debugging in interactive mode, and usually only when you want to create a file list, or when problems are

encountered while allocating the files. Refer to Appendix A, "Using the File Allocation Utility" for information about using the FAU.

The file allocation enhancement which provides your user the ability to point XPEDITER at the JCL needed to run a test is known as Quickstart. Quickstart eliminates the need to utilize the File Allocation Utility (FAU) for DB2.

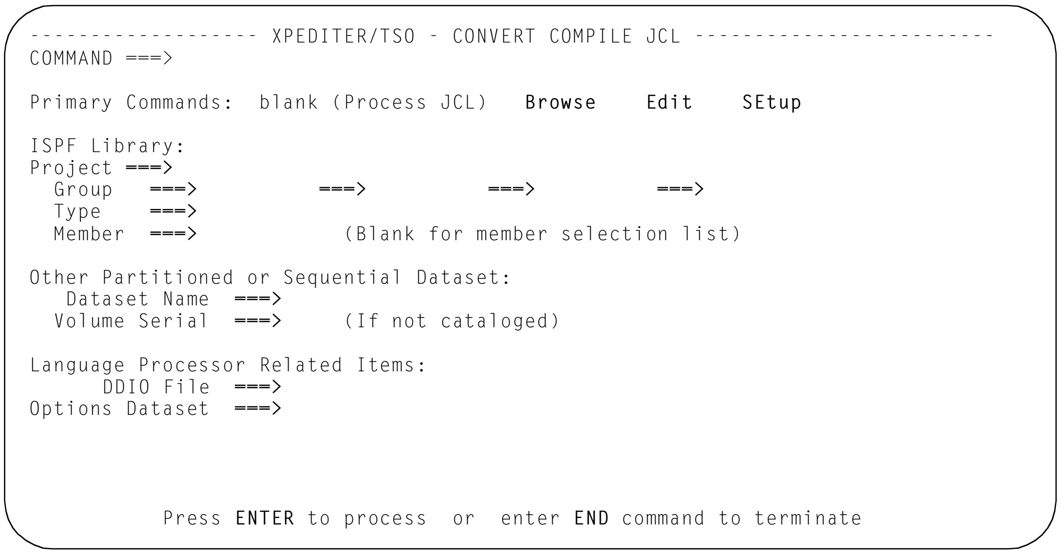
Options 1 and 2 are both compile facilities and are discussed in this chapter. Refer to the referenced appendices for information about options 3 and 4.

Converting Your Compile JCL

To automatically convert your existing compile JCL to run with the CSS language processor, enter option **1** on the Program Preparation Menu and press ENTER. The Convert Compile JCL screen shown in Figure 4-3 on page 4-4 is then displayed.

Notes:

1. If your compile JCL is not available for conversion, use the Compile Facility (option 2 on the Program Preparation Menu), which will create the compile JCL to run with the CSS language processor. On the Compile screen, use the **Editjcl** option in the Preparation field.
2. A JCL error may occur when converting a compile JCL (PROC) if a symbol is used on the XCOMPILE DD.



```
------------------ XPEDITER/TSO - CONVERT COMPILE JCL ------------------------
COMMAND ===>

Primary Commands:  blank (Process JCL)   Browse    Edit    SEtup

ISPF Library:
Project ===>
  Group   ===>          ===>          ===>          ===>
  Type    ===>
  Member  ===>            (Blank for member selection list)

Other Partitioned or Sequential Dataset:
   Dataset Name  ===>
  Volume Serial  ===>     (If not cataloged)

Language Processor Related Items:
      DDIO File  ===>
Options Dataset  ===>



           Press ENTER to process  or  enter END command to terminate
```

Figure 4-3. Convert Compile JCL Screen

In the ISPF Library field or the Other Partitioned or Sequential Dataset field, type the name of the dataset containing the JCL normally used to compile programs.

In the DDIO File field, type the name of the DDIO file into which the source listing will be placed. If you do not have a DDIO file, one can be created through the DDIO File Facility. Refer to the "DDIO File Facility" on page E-3.

In the Options Dataset field, type the name of the dataset containing LP options. If no dataset is specified, system defaults are used. LP options are general compile time options used by LP to control the source listing output for each language.

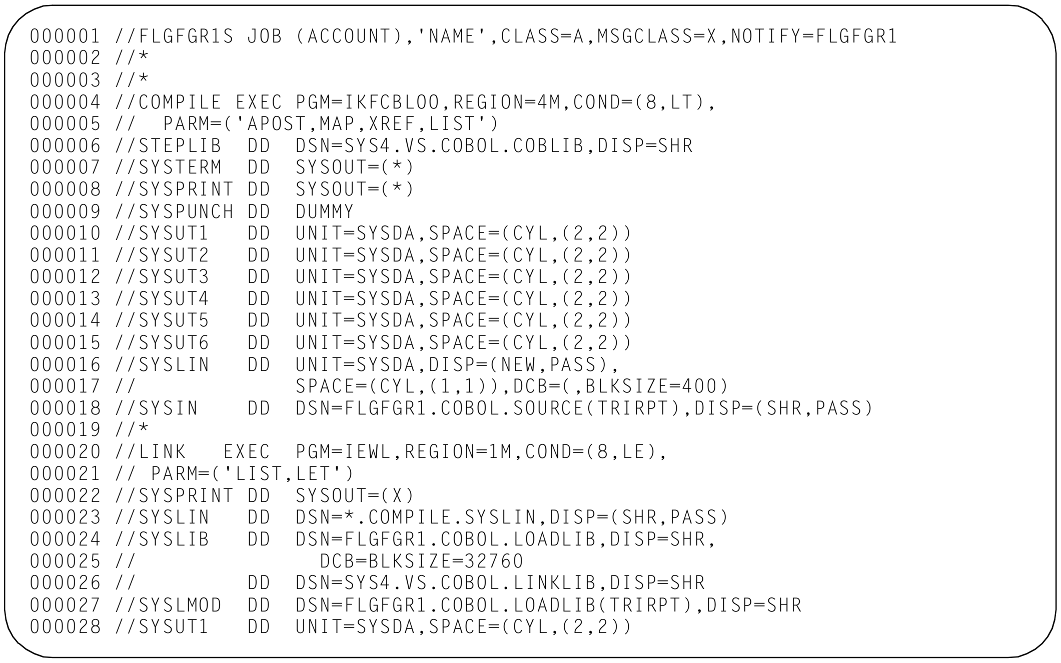
After completing the screen, you can:

- Use the BROWSE or EDIT command to display the JCL for viewing or editing.
- Use the SETUP command to set up the job card for JCL expansion.

When you press **Enter**, the JCL is submitted for conversion. If the conversion operates successfully, the converted JCL is displayed as shown in Figure 4-5 on page 4-8, with the message `JCL HAS BEEN MODIFIED TO COMPILE WITH XPEDITER`. From this screen, you can submit the job with the RUN or SUBMIT commands or use the END command to return to the previous screen.

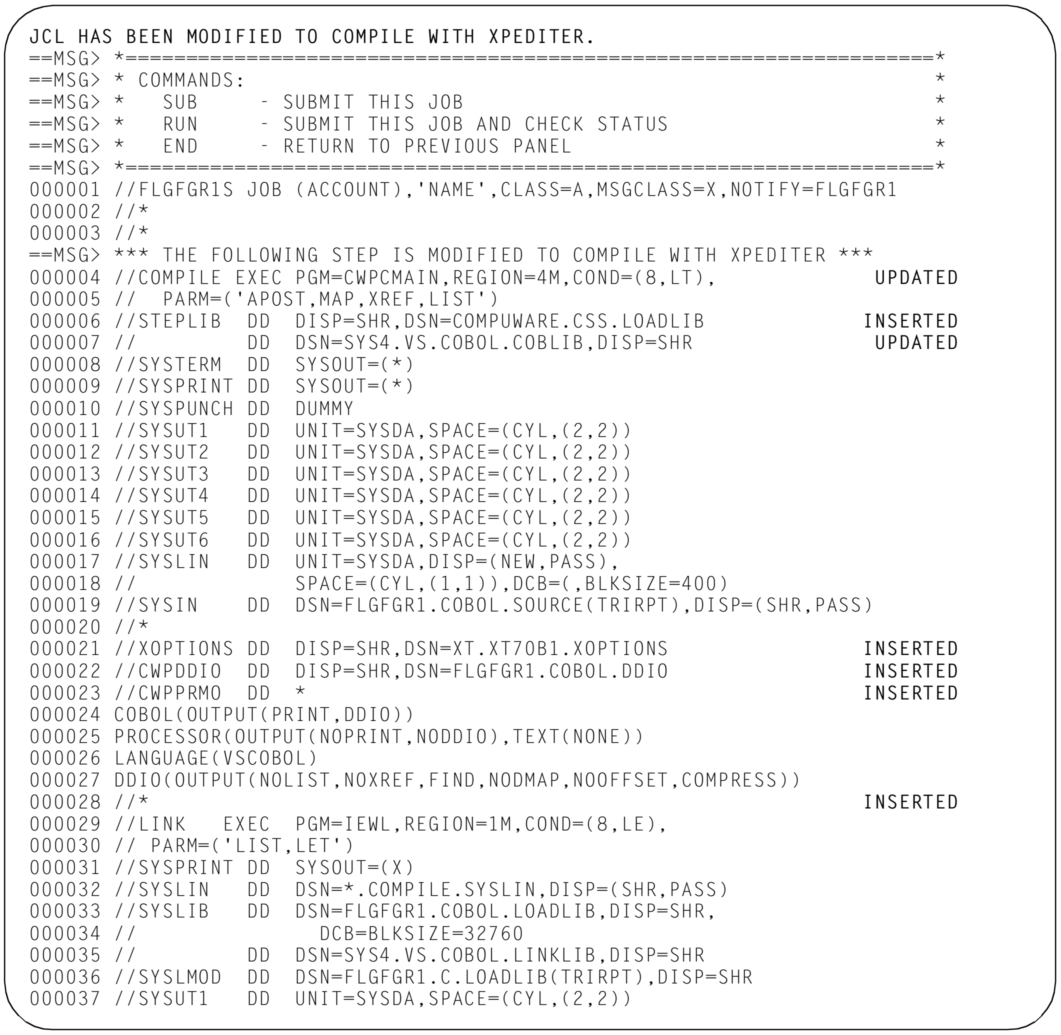
Note: If you return to the previous screen, the converted JCL is not saved.

An example of the JCL before conversion is shown in Figure 4-4 on page 4-7, and an example of the JCL after conversion is shown in Figure 4-5 on page 4-8.



```
//FLGFGR1S JOB (ACCOUNT),'NAME',CLASS=A,MSGCLASS=X,NOTIFY=FLGFGR1
//*
//*
//COMPILE EXEC PGM=IKFCBL00,REGION=4M,COND=(8,LT),
//  PARM=('APOST,MAP,XREF,LIST')
//STEPLIB  DD  DSN=SYS4.VS.COBOL.COBLIB,DISP=SHR
//SYSTERM  DD  SYSOUT=(*)
//SYSPRINT DD  SYSOUT=(*)
//SYSPUNCH DD  DUMMY
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(2,2))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(CYL,(2,2))
//SYSUT3   DD  UNIT=SYSDA,SPACE=(CYL,(2,2))
//SYSUT4   DD  UNIT=SYSDA,SPACE=(CYL,(2,2))
//SYSUT5   DD  UNIT=SYSDA,SPACE=(CYL,(2,2))
//SYSUT6   DD  UNIT=SYSDA,SPACE=(CYL,(2,2))
//SYSLIN   DD  UNIT=SYSDA,DISP=(NEW,PASS),
//             SPACE=(CYL,(1,1)),DCB=(,BLKSIZE=400)
//SYSIN    DD  DSN=FLGFGR1.COBOL.SOURCE(TRIRPT),DISP=(SHR,PASS)
//*
//LINK   EXEC  PGM=IEWL,REGION=1M,COND=(8,LE),
// PARM=('LIST,LET')
//SYSPRINT DD  SYSOUT=(X)
//SYSLIN   DD  DSN=*.COMPILE.SYSLIN,DISP=(SHR,PASS)
//SYSLIB   DD  DSN=FLGFGR1.COBOL.LOADLIB,DISP=SHR,
//               DCB=BLKSIZE=32760
//         DD  DSN=SYS4.VS.COBOL.LINKLIB,DISP=SHR
//SYSLMOD  DD  DSN=FLGFGR1.COBOL.LOADLIB(TRIRPT),DISP=SHR
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(2,2))
```

Figure 4-4. Compile JCL Before Conversion



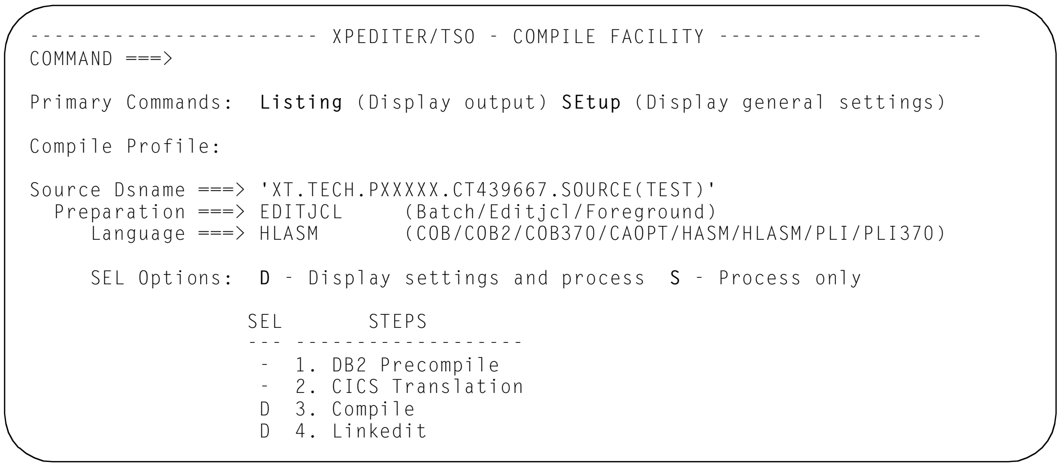
```
JCL HAS BEEN MODIFIED TO COMPILE WITH XPEDITER.
==MSG> *=================================================================*
==MSG> * COMMANDS:                                                       *
==MSG> *   SUB     - SUBMIT THIS JOB                                     *
==MSG> *   RUN     - SUBMIT THIS JOB AND CHECK STATUS                    *
==MSG> *   END     - RETURN TO PREVIOUS PANEL                            *
==MSG> *=================================================================*
000001 //FLGFGR1S JOB (ACCOUNT),'NAME',CLASS=A,MSGCLASS=X,NOTIFY=FLGFGR1
000002 //*
000003 //*
==MSG> *** THE FOLLOWING STEP IS MODIFIED TO COMPILE WITH XPEDITER ***
000004 //COMPILE EXEC PGM=CWPCMAIN,REGION=4M,COND=(8,LT),            UPDATED
000005 //  PARM=('APOST,MAP,XREF,LIST')
000006 //STEPLIB  DD  DISP=SHR,DSN=COMPUWARE.CSS.LOADLIB            INSERTED
000007 //         DD  DSN=SYS4.VS.COBOL.COBLIB,DISP=SHR               UPDATED
000008 //SYSTERM  DD  SYSOUT=(*)
000009 //SYSPRINT DD  SYSOUT=(*)
000010 //SYSPUNCH DD  DUMMY
000011 //SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(2,2))
000012 //SYSUT2   DD  UNIT=SYSDA,SPACE=(CYL,(2,2))
000013 //SYSUT3   DD  UNIT=SYSDA,SPACE=(CYL,(2,2))
000014 //SYSUT4   DD  UNIT=SYSDA,SPACE=(CYL,(2,2))
000015 //SYSUT5   DD  UNIT=SYSDA,SPACE=(CYL,(2,2))
000016 //SYSUT6   DD  UNIT=SYSDA,SPACE=(CYL,(2,2))
000017 //SYSLIN   DD  UNIT=SYSDA,DISP=(NEW,PASS),
000018 //             SPACE=(CYL,(1,1)),DCB=(,BLKSIZE=400)
000019 //SYSIN    DD  DSN=FLGFGR1.COBOL.SOURCE(TRIRPT),DISP=(SHR,PASS)
000020 //*
000021 //XOPTIONS DD  DISP=SHR,DSN=XT.XT70B1.XOPTIONS               INSERTED
000022 //CWPDDIO  DD  DISP=SHR,DSN=FLGFGR1.COBOL.DDIO               INSERTED
000023 //CWPPRMO  DD  *                                             INSERTED
000024 COBOL(OUTPUT(PRINT,DDIO))
000025 PROCESSOR(OUTPUT(NOPRINT,NODDIO),TEXT(NONE))
000026 LANGUAGE(VSCOBOL)
000027 DDIO(OUTPUT(NOLIST,NOXREF,FIND,NODMAP,NOOFFSET,COMPRESS))
000028 //*                                                         INSERTED
000029 //LINK   EXEC  PGM=IEWL,REGION=1M,COND=(8,LE),
000030 // PARM=('LIST,LET')
000031 //SYSPRINT DD  SYSOUT=(X)
000032 //SYSLIN   DD  DSN=*.COMPILE.SYSLIN,DISP=(SHR,PASS)
000033 //SYSLIB   DD  DSN=FLGFGR1.COBOL.LOADLIB,DISP=SHR,
000034 //               DCB=BLKSIZE=32760
000035 //         DD  DSN=SYS4.VS.COBOL.LINKLIB,DISP=SHR
000036 //SYSLMOD  DD  DSN=FLGFGR1.C.LOADLIB(TRIRPT),DISP=SHR
000037 //SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(2,2))
```

Figure 4-5. Compile JCL After Conversion

Compiling Your Programs

To access the compile facility, type option **2** (COMPILE FACILITY) on the Program Preparation Menu. The Compile Facility screen shown in Figure 4-6 on page 4-9 is displayed.

On this screen, you will specify what you want to do (DB2 precompile, translate CICS commands, compile and/or link edit), the name of the dataset to be used, how the job is to be run, and the program language. Depending on your input, you are presented with additional screens where you can enter more detailed information.



```
----------------------- XPEDITER/TSO - COMPILE FACILITY ---------------------
COMMAND ===>

Primary Commands:  Listing (Display output) SEtup (Display general settings)

Compile Profile:

Source Dsname ===> 'XT.TECH.PXXXXX.CT439667.SOURCE(TEST)'
  Preparation ===> EDITJCL     (Batch/Editjcl/Foreground)
     Language ===> HLASM       (COB/COB2/COB370/CAOPT/HASM/HLASM/PLI/PLI370)

     SEL Options:  D - Display settings and process  S - Process only

                  SEL       STEPS
                  --- -------------------
                   -  1. DB2 Precompile
                   -  2. CICS Translation
                   D  3. Compile
                   D  4. Linkedit
```

Figure 4-6. Compile Facility Screen

Compile Profile

The XPEDITER Compile Facility provides for separate compile profiles. This allows you to save the input dataset name, compile options, compile DDIO, compile LOADLIB name, and link options in different profiles. You are given the flexibility to name the profiles in any manner you choose. This permits you to use program names, generic compile processes, or special compile group names.

To maintain these profiles, you must enter a '?' in the profile name when entering the compile facility. Doing this will cause a list of the current compile profiles to appear on your screen for your analysis. At this point, you can Delete, Copy, Rename, and Select different profiles.

Source Dsname

Type the name of the dataset containing the program to be processed. Type the object dataset name for link-edit only.

Note: If you do not enter a member name in this field, a member list is displayed. Selecting a member name on the member list screen automatically builds the JCL and submits the job if Batch is specified in the Preparation field. If Editjcl is specified, the JCL is built and you can edit the JCL before submitting the job. If Foreground is specified, the program is automatically compiled in foreground. As the job is processing, the third line on the member list shows the status of the job. When the job completes, the return code is displayed next to the member name. You can use the **L** (Listing) line command to view the source listing associated with the member.

Preparation

Specify one of the following:

Batch	Builds the JCL and submits the JCL to compile in batch.
Editjcl	Builds the JCL and allows editing of the JCL after it is built. The JCL can be submitted from the Edit screen.
Foreground	Compiles the program in foreground.

Language

Enter the program language type.

SEL Options

Enter the SEL options in the SEL field to specify the actions to be performed (DB2 precompile, translate CICS commands, compile, and/or link-edit). The SEL options are:

D Displays the settings screen for the action. The settings screen is used to provide additional information about the action you are performing. When you press the **Enter** key, the settings screen for the first action you selected is displayed. When you press **Enter** on the displayed settings screen, the settings screen for the next selected action is displayed, and so forth.

S Processes the selected action without displaying the settings screen.

The primary commands are used as follows:

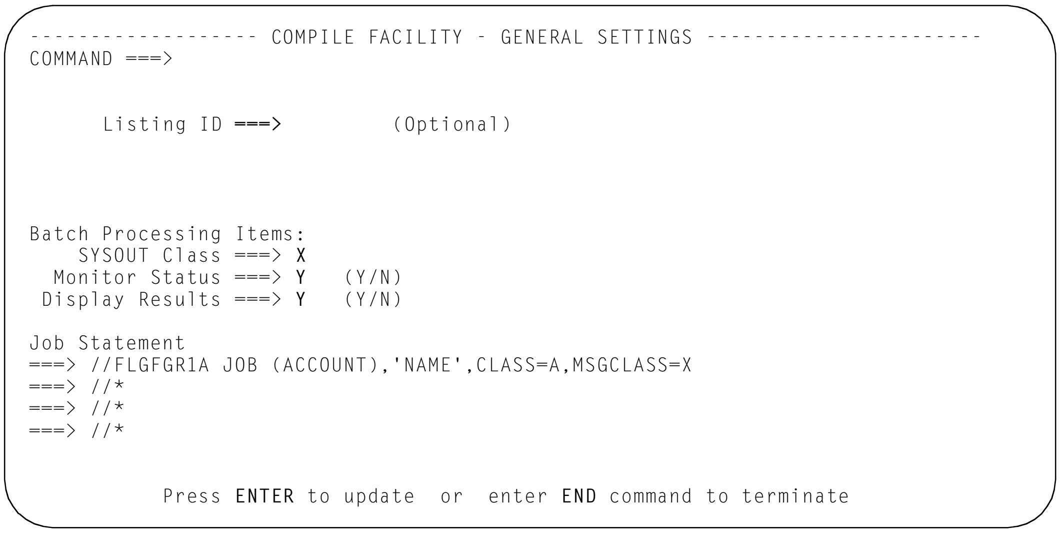
Listing
Displays the compile listing output.

SEtup
Displays the General Settings screen on which you can override the default general settings, such as listing ID, SYSOUT class, job card information, and so on.

The General Setting screen and the settings screens for each option are described below.

General Settings Screen

The General Settings screen shown in Figure 4-7 is used to override the default settings for processing a job.



```
------------------- COMPILE FACILITY - GENERAL SETTINGS -----------------------
COMMAND ===>


      Listing ID ===>         (Optional)




Batch Processing Items:
    SYSOUT Class ===> X
  Monitor Status ===> Y   (Y/N)
 Display Results ===> Y   (Y/N)

Job Statement
===> //FLGFGR1A JOB (ACCOUNT),'NAME',CLASS=A,MSGCLASS=X
===> //*
===> //*
===> //*


           Press ENTER to update  or  enter END command to terminate
```

Figure 4-7. General Settings Screen

Listing ID
Type the dataset name for the source listing output. If blank, the listing output is directed to SYSOUT for batch processing and to *prefix*.MEMBER.*list* for foreground processing.

SYSOUT Class
Specify the SYSOUT class to which output is routed.

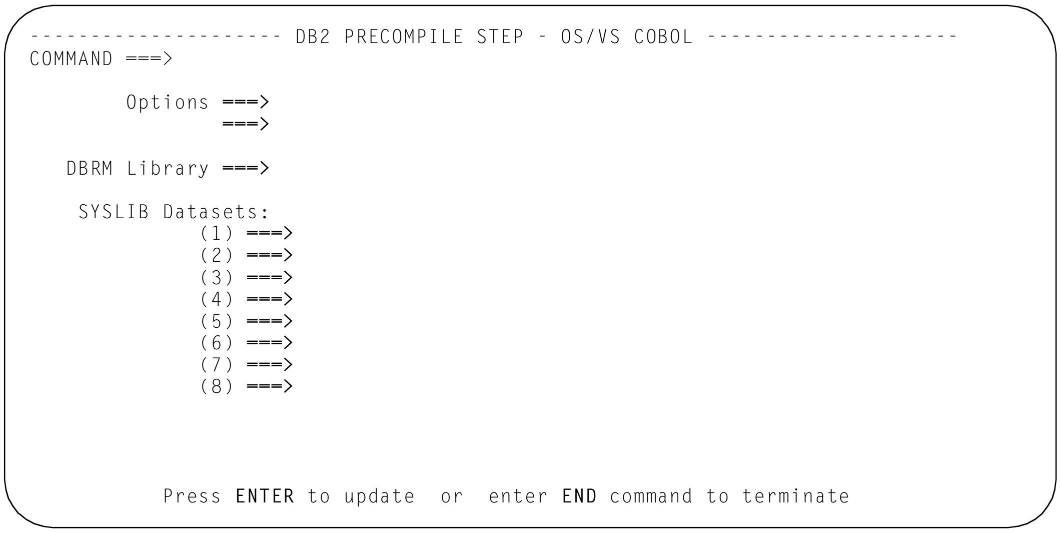
Monitor Status
Specify whether the status of the job should be monitored. If **Y** is entered, the Submit Status screen is displayed. The **Attn** key can be used to cancel monitoring at any time.

Display Results
Specify whether the results of the steps should be displayed after the job completes.

Job Statement
Specify job card information if the execution environment is batch or Editjcl. This information is optional if your site has a submit exit.

DB2 Precompile Step Screen

The DB2 Precompile Step screen shown in Figure 4-8 is used to specify the settings and options for the DB2 precompile.



```
-------------------- DB2 PRECOMPILE STEP - OS/VS COBOL --------------------
COMMAND ===>

        Options ===>
                ===>

   DBRM Library ===>

    SYSLIB Datasets:
              (1) ===>
              (2) ===>
              (3) ===>
              (4) ===>
              (5) ===>
              (6) ===>
              (7) ===>
              (8) ===>



           Press ENTER to update  or  enter END command to terminate
```

Figure 4-8. DB2 Precompile Step Screen

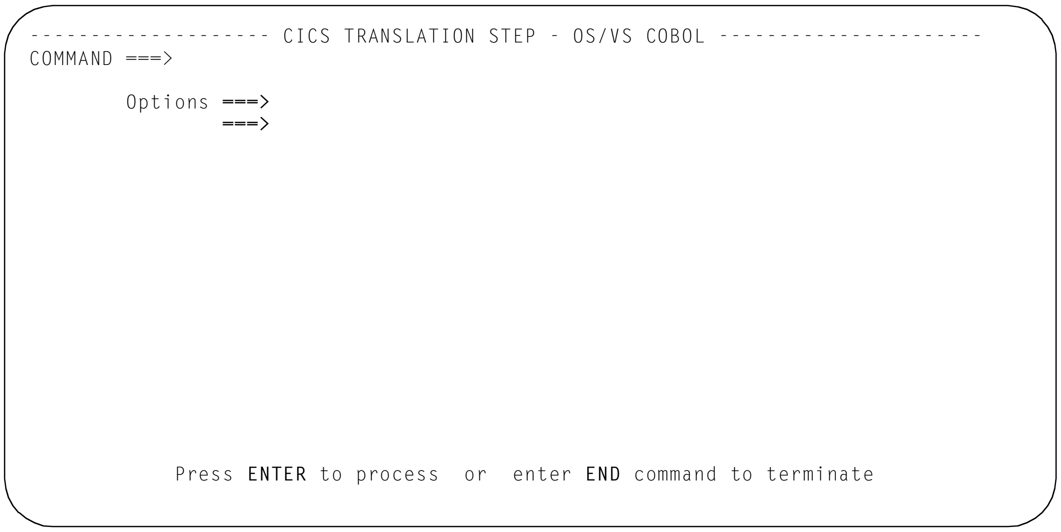
Options
Enter the DB2 precompile options. If a parameter cannot fit on the first line, use the second line. For information about these options, refer to the appropriate IBM manual.

DBRM library
Type the dataset name to which the precompiled output is written. This dataset is the input to the bind process. If no dataset name is specified, a default is created with DSORG=PO.

SYSLIB Datasets
Enter the names of the datasets to be used as secondary input. These datasets must be partitioned, RECFM=F or FB, and LRECL=80.

CICS Translation Step Screen

The CICS Translation Step screen shown in Figure 4-9 is used to specify the options for EXEC CICS and EXEC DLI command translation.



```
------------------- CICS TRANSLATION STEP - OS/VS COBOL ---------------------
COMMAND ===>

        Options ===>
                ===>



            Press ENTER to process  or  enter END command to terminate
```

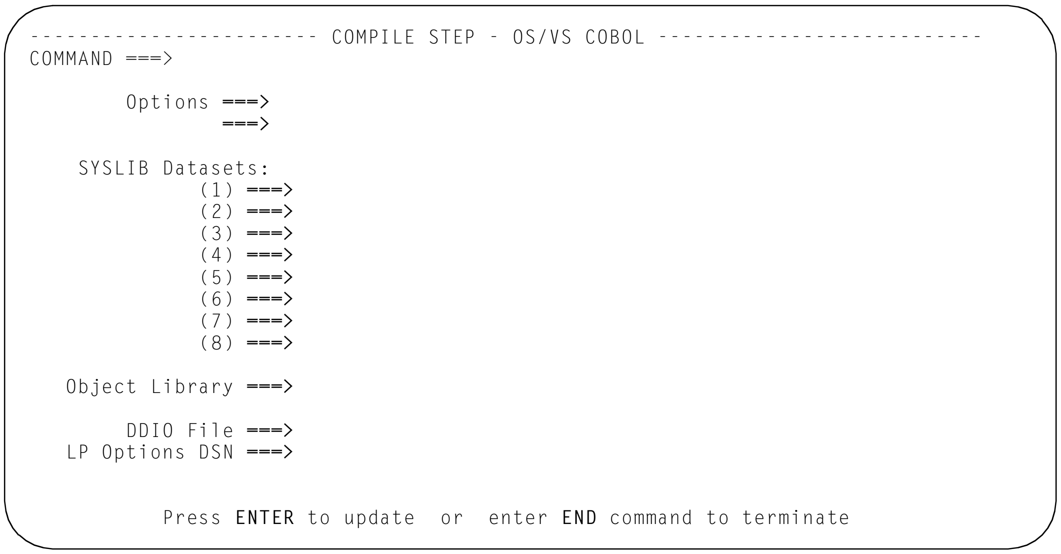
Figure 4-9. CICS Translation Step Screen

Options

Enter the CICS command translation input/output parameter options. If a parameter cannot fit on the first line, use the second line. Refer to the appropriate IBM manual for information about the options.

Compile Step Screen

The Compile Step screen shown in Figure 4-10 is used to specify the settings and options for the compile.



```
----------------------- COMPILE STEP - OS/VS COBOL --------------------------
COMMAND ===>

        Options ===>
                ===>

    SYSLIB Datasets:
              (1) ===>
              (2) ===>
              (3) ===>
              (4) ===>
              (5) ===>
              (6) ===>
              (7) ===>
              (8) ===>

   Object Library ===>

        DDIO File ===>
   LP Options DSN ===>


           Press ENTER to update  or  enter END command to terminate
```

Figure 4-10. Compile Step Screen

Options

Enter the compile parameter options. If a parameter option cannot fit on the first line, use the second line. Refer to the appropriate IBM language manual for information about the options.

XPEDITER requires specific compile options which may vary depending on the language type and actual version being used. A user should always refer to the appropriate *CSS User/Reference Guide* for information related to any required compiler options. Examples are: (1) Options NOTEST and NONUMBER are required for COBOL; (2) Option NOTEST is required for PL/I, and (3) Options ESD and LIST are required for Assembler. The *guide* is required to assist users in manually converting Compile and Link JCL to execute the CSS Language Processor. The *guide* also gives most users information on how to set up the Language Processor parameters to write a matching source listing to a DDIO file.

SYSLIB Datasets

Type the names of the datasets to be used for copy and copybook processing. A maximum of 8 names can be specified.

Statistics

Type the name of the CA-Optimizer statistics dataset. This field only appears when CAOPT is specified for the application language.

Object Library

Optional. Specify the object library name. If blank, a temporary dataset is used.

DDIO File

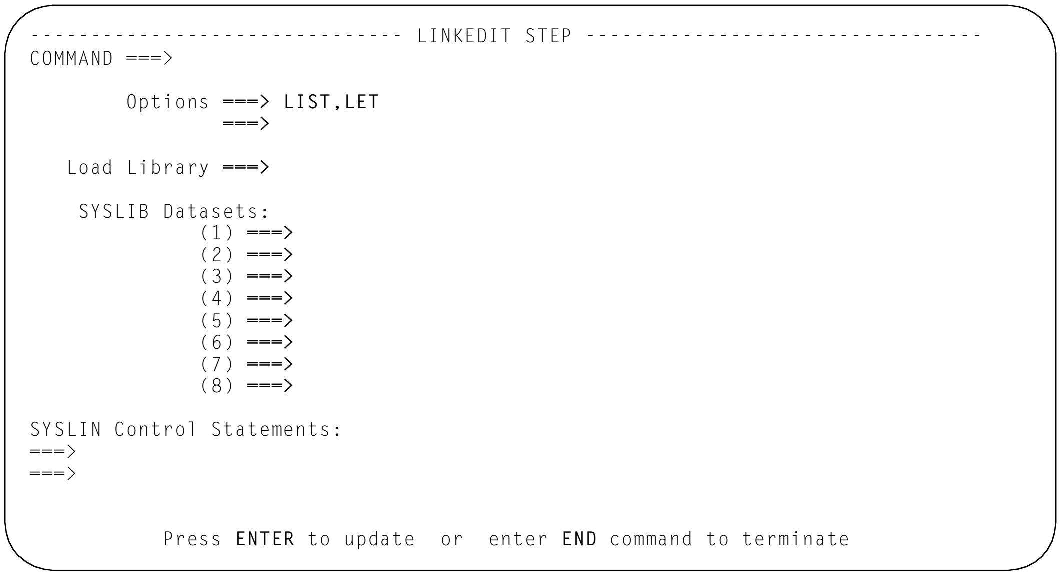
Type the name of the DDIO file to which the source listing is written after the compile is complete. If blank, the system prompts you to create a DDIO file. Refer to the "DDIO File Facility" on page E-3 for information about creating this file.

LP Options DSN
Optional. Type the name of the user dataset containing the language processor options. If blank, the system defaults are used.

Note: Language processor options are general compile time options used by the LP to control source listing output for all languages. Contact your local technical support.

Linkedit Step Screen

The Linkedit Step screen shown in Figure 4-11 is used to specify the settings and options for the link-edit.



```
------------------------------ LINKEDIT STEP --------------------------------
COMMAND ===>

        Options ===> LIST,LET
                ===>

   Load Library ===>

    SYSLIB Datasets:
              (1) ===>
              (2) ===>
              (3) ===>
              (4) ===>
              (5) ===>
              (6) ===>
              (7) ===>
              (8) ===>

SYSLIN Control Statements:
===>
===>


           Press ENTER to update  or  enter END command to terminate
```

Figure 4-11. Linkedit Screen

Options
Enter the linkage editor parameter options. If a parameter cannot fit on the first line, use the second line. All link-edit options are supported except OVLY and NE. Refer to the appropriate IBM manual for more information about these options.

Load Library
Specify the name of a partitioned dataset to contain the load module after the link-edit is completed. If blank, the system will generate a dataset name that follows the dataset naming conventions.

SYSLIB Datasets
Enter the names of the datasets to be searched by the linkage editor to locate object modules referenced by the module being processed. A maximum of 8 names can be specified.

SYSLIN Control Statements
Specify the control card statements in this field.

Starting an Interactive Session

To start an interactive debugging session, do the following:

1. Access the XPEDITER/TSO Primary Menu.

   Note: If you want to change the defaults for your terminal, PF keys, screen colors, profile, etc., use option **0**. Refer to Appendix D, “Specifying Session Defaults” for more information.

2. Optionally, specify a new or existing profile for the debugging session. When a profile is not specified, a default profile is used. Refer to “Using Profiles” on page 4-13.
3. Use option **2** (TSO) on the XPEDITER/TSO Primary Menu. Secondary screens are displayed for specifying the program name and information about the environment in which the program will execute.

   The SETUP command shown on some of the secondary screens can be used to override the installed setup options, such as the name of the load library and DDIO dataset, log disposition, and so on. The Setup Facility is discussed in detail in Appendix C, “Specifying Setup Options”.
4. Allocate the files and databases that your program will need to execute. See “Allocating the Required Files” on page 4-15.

Using Profiles

The information you specify for a debugging session is recorded in a user profile that can be used each time you want to execute the same test session again. This information includes the name of the program, the list of files and databases that must be allocated for the program to execute, the setup options for the environment in which the program will execute, and the XPEDITER/TSO defaults to be associated with the test session. It also contains the defaults established by your systems programming staff during installation of XPEDITER/TSO.

The profile identifier and description are displayed in the Profile field on the Primary Menu. The Profile field can be used to:

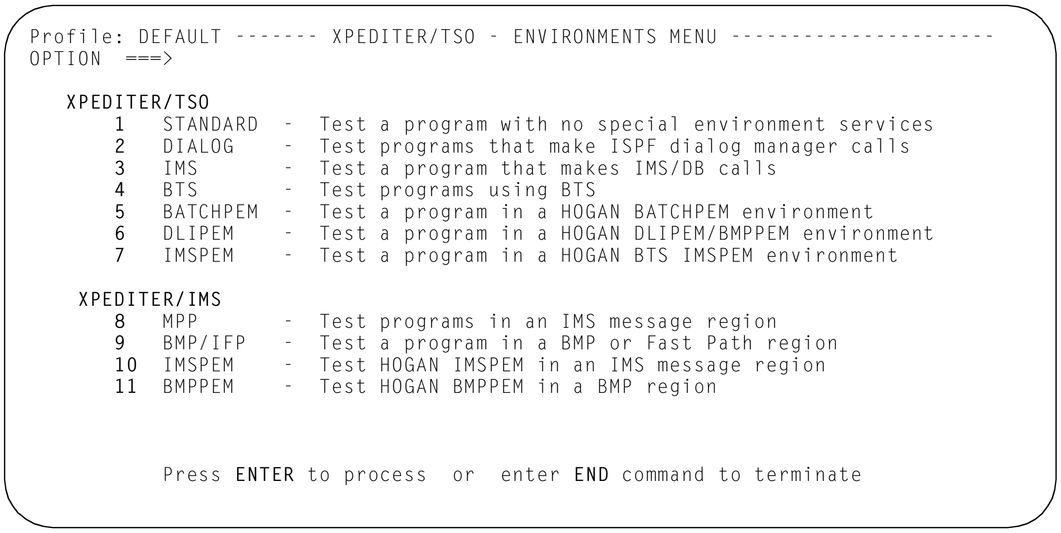
- Create a New Profile
- Use an Existing Profile
- Obtain a List of Profiles

Changes to a user profile are made through the Profile screen shown in Figure D-5 on page D-5. This screen can also be accessed through option **0** (Defaults) on the Primary Menu or by entering the **PROFILE** command from the Primary Menu screen or any test screen. See “Specifying User Profiles” on page D-4 for information about the Profile facility.

Invoking the Test Session

The first time you invoke an XPEDITER test session using option 2 (TSO) on the Primary Menu, the Environments Menu shown in Figure 4-12 on page 4-14 is displayed.

Note: Subsequent invocations of XPEDITER/TSO will display the last environment test screen that was used. The Environments Menu is only displayed again when you are creating a new profile or when you access it through the setup facility using the SETUP command. The SETUP command is discussed briefly later in this section and in more detail in Appendix C, “Specifying Setup Options”.



```
Profile: DEFAULT ------- XPEDITER/TSO - ENVIRONMENTS MENU ---------------------
OPTION  ===>

   XPEDITER/TSO
       1   STANDARD  -  Test a program with no special environment services
       2   DIALOG    -  Test programs that make ISPF dialog manager calls
       3   IMS       -  Test a program that makes IMS/DB calls
       4   BTS       -  Test programs using BTS
       5   BATCHPEM  -  Test a program in a HOGAN BATCHPEM environment
       6   DLIPEM    -  Test a program in a HOGAN DLIPEM/BMPPEM environment
       7   IMSPEM    -  Test a program in a HOGAN BTS IMSPEM environment

    XPEDITER/IMS
       8   MPP       -  Test programs in an IMS message region
       9   BMP/IFP   -  Test a program in a BMP or Fast Path region
       10  IMSPEM    -  Test HOGAN IMSPEM in an IMS message region
       11  BMPPEM    -  Test HOGAN BMPPEM in a BMP region



           Press ENTER to process  or  enter END command to terminate
```

Figure 4-12. Environments Menu

The range of environment options shown on this menu depends on the site defaults set by your installer. More information about the environments is provided in “Guidelines for Selecting an Environment” on page 4-14.

Each option on the Environments menu accesses an environment test screen on which you enter information about your program and the test session. For example, if you are testing a program in the standard environment with no special services, you would type option **1** (STANDARD) on the Environments Menu and the Standard test screen would be displayed.

Environment test screens are used to specify all of the information required about the program and test session. All environment test screens are described in Appendix B, “XPEDITER/TSO Environment Test Screens”.

If you type **SETUP** on the command line of the environment test screen, the Setup Menu is displayed. The options on the Setup Menu provide additional screens on which you can change the installed values for the listed options, such as load library names, DDIO file name, log disposition, and so on. All setup menus and screens are described in Appendix C, “Specifying Setup Options”.

When you complete the test screen and specify all setup information, an XPEDITER/TSO test session is started. Your source is displayed on the XPEDITER/TSO source display where all debugging functions are available for your use. The debugging functions are discussed in Chapter 5, “Debugging Interactively”.

Guidelines for Selecting an Environment

The XPEDITER/TSO options (1 through 7) on the Environments Menu displayed in Figure 4-12 invoke a debugging session and execute the application program in the TSO address space. The XPEDITER/IMS options (8 through 11) on the Environments Menu invoke a debugging session and execute the IMS/DC application program in the message region, the BMP region, and the Fast Path region.

If you are developing IMS/DC applications, XPEDITER/IMS provides the capability of debugging IMS/DC programs that are actually running in the IMS dependent regions, whereas XPEDITER/TSO provides the capability of debugging IMS/DC applications with the use of a simulator (BTS) under TSO. Operations of XPEDITER/IMS, however, require at least one logical TSO terminal and one logical IMS terminal (it can be an ATM terminal), both on the same CPU.

The following table summarizes the environment options to choose, depending on the type of application you want to debug:

Environment Option	Type of Application Program
STANDARD	• Batch programs that process QSAM and VSAM files. • Batch programs that issue EXEC SQL (DB2) statements. • Batch programs that issue third-party database calls (IDMS/DB, ADABAS, TOTAL, TIS, SUPRA, DATACOM/DB, System 2000).
DIALOG	• Dialog applications composed of screens, CLISTs, load modules, messages, command tables, and file-tailoring skeletons. • Programs that run as part of a user-provided system involving multiple tasks.
IMS	• Batch programs that issue CBLTDLI (IMS/DB) calls. • Batch programs that issue CBLTDLI (IMS/DB) calls • IMS BMP programs. Refer to BMP/IFP.
BTS	• IMS/DC MPP programs with the use of BTS. Refer to MPP option.
BATCHPEM	• Hogan BATCHPEM applications or the BATCHPEM driver module itself.
DLIPEM	• Hogan DLIPEM applications or the DLIPEM driver module itself.
IMSPEM	• Hogan IMSPEM applications or the driver module itself with the use of BTS.
MPP	• IMS/DC MPP programs in the IMS message region. Refer to BTS option.
BMP/IFP	• IMS BMP programs. Refer to IMS option. • Fast Path programs in the IMS Fast Path region.
IMSPEM	• Hogan IMSPEM applications or the driver module itself in the IMS MPP region.
BMPPEM	• Hogan BMPPEM applications or the driver module itself in the IMS BMP region.

Table 4-1. Guidelines for Choosing an Environment.

Allocating the Required Files

This section tells you how to interactively specify a file list to be used at execution time to allocate the files and databases your program needs.

Note: All files used by your program **must** be allocated, or your program will abend as if you had omitted a DD statement from a batch run.

Allocating the files your program will need is as easy as entering the dataset name of an existing file list, a CLIST, or your execution JCL (also identified as Quickstart) in the File List/JCL Member field on the environment test screen.

There may be times when you may need to allocate files after starting a test session. This can be done with the ALLOCATE command, which can be entered from any XPE-DITER/TSO screen.

At other times, you may need to use the File Allocation Utility (FAU), a file list editor that lets you create and edit file lists. The FAU is accessed through option 1 on the XPE-DITER/TSO Primary Menu. When errors occur during file allocations while using the test screen or the ALLOCATE command, the FAU is automatically accessed to assist you in resolving the problem.

Each of these methods is used for a somewhat different purpose. The first two methods are described below. A description of the File Allocation Utility is provided in Appendix A, “Using the File Allocation Utility”.

Allocating From the Test Screen

The easiest and most commonly used way to allocate the files your program will need is to enter the name of the dataset containing the file list, CLIST, or the program execution

JCL (also identified as Quickstart) in the File List/JCL Member field on the environment test screen as shown in the following example:

```
File List/JCL Member ===> 'XT.XT70.FAU(TRIMAIN)'
```

If the dataset contains a file list or CLIST and there are no errors, the files are automatically allocated.

If the dataset contains JCL, the following occurs before the files are allocated:

1. If the JCL executes a PROC, the PROC is automatically expanded. See "Things to Know About JCL Expansion" on page A-13 for more information.
2. JCL DD statements are automatically converted to the file list format. The conversion rules are as follows:
   - SMS keywords, in-stream data, ROUND (and other space parameters), SYSOUT related parameters (HOLD, COPIES, DEST, etc.) are processed.
   - JCL statements that do not have the corresponding SVC 99 support are processed by ignoring the keywords and parameters. This situation will take place, for instance, when VSAM AMP parameters are encountered.
   - The program references EXEC PGM=*.*xxx.yyy* cannot be processed.

     Processing will pause, the FAU is accessed, and the appropriate screens that will allow you to correct the problem are displayed.
   - When the program name on the test screen does not match the EXEC PGM in the JCL, an error message is displayed and the FAU is automatically accessed.

If no errors occur, the files for the debugging session are automatically allocated. If a problem occurs, the FAU is automatically accessed, and you are presented with the screens needed to correct the problem.

Allocating With the ALLOCATE Command

The ALLOCATE command is generally used to allocate files from within an XPEDITER test session. It can be entered on the command line of any XPEDITER screen. If entered without a dataset name, the FAU is accessed and you can specify the information about the file to be allocated. When you exit the FAU, the files are automatically allocated.

If you enter the ALLOCATE command with a dataset name, the dataset is automatically accessed. If the dataset contains a file list and the allocations are successful, **you will not see any FAU facilities**.

If the dataset contains JCL, the FAU is accessed and the JCL dataset is displayed on the Select DDNAME screen. On this screen, you can select the DD statements to be copied to the file list. Refer to Appendix A, "Converting JCL to a File List" on page A-12.

If an error occurs during the allocation, the Edit File List screen is displayed with an error message. At this time, you can correct the problem. When you enter **END**, the files are allocated.

Usage Note

Even if your application programs would normally find any required Language dependent run-time subroutines (including LE - Language Environment), without being included in the JOBLIB/STEPLIB of the batch JCL (usually from the LINKLIST or (E)LPA), the libraries must still be specified as part of the test session setup. This will ensure that XPEDITER's Task Library will be properly configured.

Using XPEDITER/TSO Commands in Interactive Mode

In interactive mode, debugging is performed by entering XPEDITER/TSO commands online.

You can also use script files in interactive mode.

- An initial script member name can be specified in the Initial Script field on the selected environment test screen.
- A test script member name can be specified by using the INCLUDE command on the command line during the test session.
- A post script member name can be specified in the Post Script field on the selected environment test screen.
- An abend script is not applicable in interactive mode.

For information about using scripts, refer to “Test Session Management Using Scripts” on page 4-42.

Starting a Batch Connect Session

With the batch connect facility, you can submit your execution JCL in batch and connect directly to the job as it executes in its native environment—the MVS batch initiator.

Notes:

1. In batch connect mode, you do not have to allocate the files and databases required for your program to run—they are automatically allocated to the batch address space via the DD statements in the JCL.
2. No change in the TSO logon size is required for large programs.
3. Batch connect is also available for the ROSCOE environment. See “Batch Connect Under ROSCOE” on page 4-26.
4. The batch connect facility does not use ISPF services.
5. Batch connect does not support nested procedures.
6. The CICS interface to DLI driver (DFHDRP) is not supported in batch connect.

The batch connect facility presents several screens that are used to automate the process of starting the session. The following summarizes the process:

1. **Access the Facility**: Type option **3** (BATCH) on the XPEDITER/TSO Primary Menu to access the batch connect facility. The Process Execute JCL screen is displayed.
2. **Process the execution JCL**: The Process Execute JCL screen is used to identify the execution JCL for the debugging session. You can also access the ISPF editor to view and make changes to the JCL.

   The JCL statements are scanned, expanded if the JCL is a procedure, and the job steps are extracted. A list of the job steps is then displayed on the Select Job Step screen.

   The following note is directed to COBOL users only:

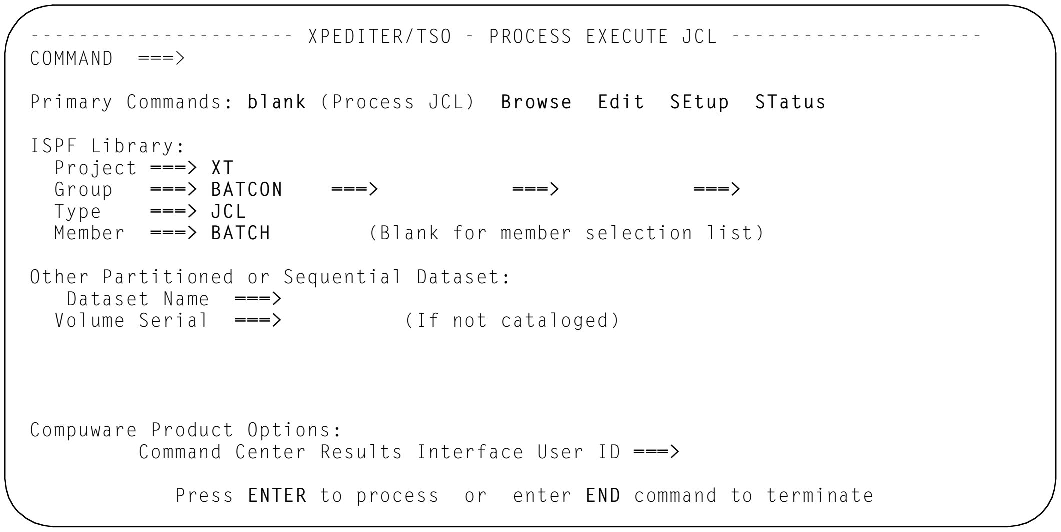
   Note: As part of the process to retrieve the job steps of the expanded procedure, XPEDITER inserts a "**/*ROUTE PRINT LOCAL**" card into a JCL job stream. The option to *"insert"* or *"not insert"* the card is controlled by the CTLROUTE option in the JCLRA093 member. Refer to the *XPEDITER/TSO and XPEDITER/IMS Installation Guide*, Appendix G (Optional Postinstallation Procedures), under the heading 'ESTABLISHING SITE DEFAULTS - CUSTOMIZING THE RUN-TIME ENVIRONMENT', for more information.
3. **Convert and Submit the JCL**: The Select Job Step screen is used to specify how you want each job step to execute (in interactive or unattended mode). When a job step is selected for interactive debugging, the source for that step is displayed at your terminal. When a step is selected to run in unattended mode, you cannot interact with the step from your terminal. If you want to debug the step, XPEDITER debugging commands must be read from a test script and the output results from the test session are written to the log.

You can submit the job from this screen or you can edit the JCL. When you submit the job, the batch connect intelligent scanner automatically converts each step to execute in the specified mode (interactive or unattended). See "JCL Conversion" on page 4-24 for information about JCL conversion.

4. **Connect to the Job**: If you submit the job with the RUN command, you are automatically connected to the job steps that were specified to execute in interactive mode. If you submit the job with the SUBMIT command, you must use the CONNECT or STATUS command to connect your terminal to the job steps selected for interactive debugging.
5. **Edit the JCL**: Optionally, you can access the edit facility from the Process Execute JCL or Select Job Step screen to view and confirm the JCL conversion or to make changes to the JCL.
6. **Access the Setup Facility**: Optionally, you can use the SETUP command on the Process Execute JCL screen or the Select Job Step screen to access the Setup Menu. From the Setup Menu, you can view and change your job card information and change the installed defaults, such as the load libraries, DDIO dataset, and so on. Refer to Appendix C, "Specifying Setup Options" for a description of the setup screens.

Processing the JCL

The Process Execute JCL screen shown in Figure 4-13 is used to specify the name of the JCL to be used for the debugging session and to specify what you want to do with the JCL.



```
--------------------- XPEDITER/TSO - PROCESS EXECUTE JCL ---------------------
COMMAND  ===>

Primary Commands: blank (Process JCL)  Browse  Edit  SEtup  STatus

ISPF Library:
  Project ===> XT
  Group   ===> BATCON    ===>          ===>          ===>
  Type    ===> JCL
  Member  ===> BATCH        (Blank for member selection list)

Other Partitioned or Sequential Dataset:
   Dataset Name  ===>
  Volume Serial  ===>          (If not cataloged)



Compuware Product Options:
         Command Center Results Interface User ID ===>

            Press ENTER to process  or  enter END command to terminate
```

Figure 4-13. Process Execute JCL Screen

There are two things you need to do on this screen:

1. Enter the name of the dataset containing your execution JCL either in the ISPF Library field or in the Other Partitioned or Sequential Dataset field.
2. Use one of the following commands on the command line:

   blank
   Leave the command line blank to scan the specified JCL and extract the job step information. If the JCL is a procedure, it is expanded prior to scanning. A progress message is displayed during processing. When processing is complete, the Select Job Step screen shown in Figure 4-14 on page 4-19 is displayed.

   Browse
   Invokes the ISPF browse facility and displays the specified JCL.

Edit
Invokes the ISPF edit facility and displays the specified JCL. You can edit the JCL. Refer to “Editing the JCL” on page 4-22 for more information.

SEtup
Accesses the Test Setup Menu from which you can access the job card information needed to process JCL PROCs. Refer to Appendix C, “Specifying Setup Options” for more information.

STatus
Displays the status of any submitted job(s). You can connect to a job from the Status screen.

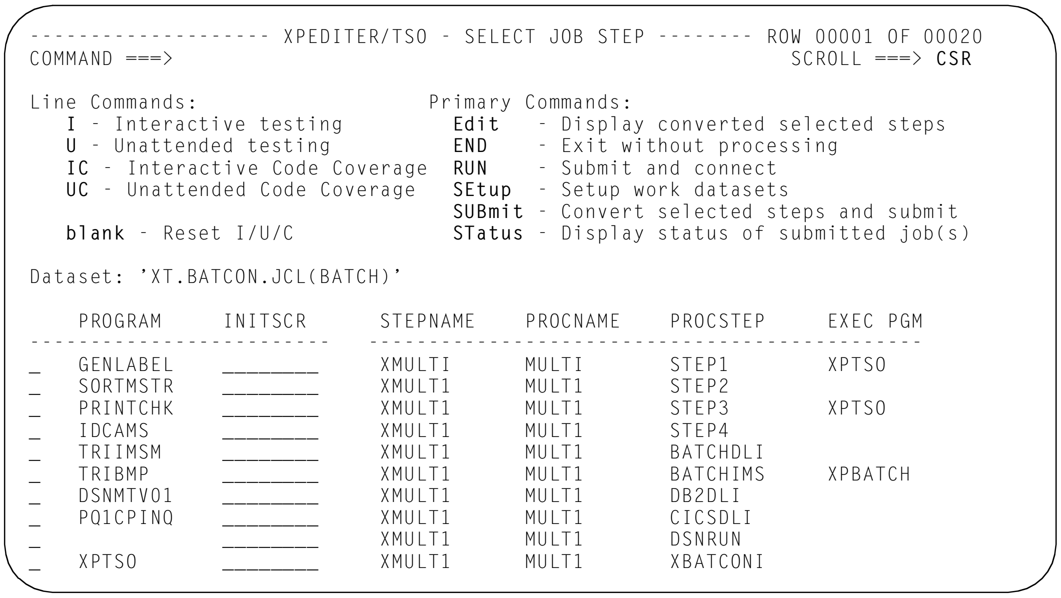
The Compuware Product Options section is used for the following:

Note: If the install option for this product is set to NO, this area will not appear.

Command Center Results Interface User ID
Enter the Command Center user ID if you are accessing the batch connect facility from a TSO session under Command Center. The Results Interface pop-up window will be opened at the time the job is ready to connect for interactive testing.

Converting and Submitting the JCL

When JCL processing is completed, the job steps in the specified JCL are displayed on the Select Job Step screen shown in Figure 4-14.



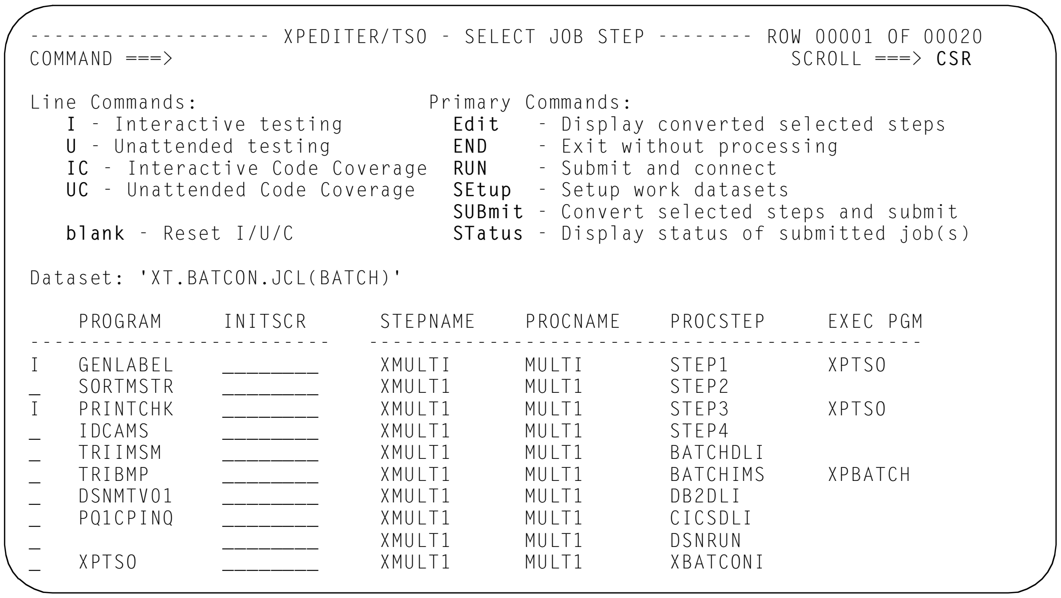
```
------------------- XPEDITER/TSO - SELECT JOB STEP -------- ROW 00001 OF 00020
COMMAND ===>                                                 SCROLL ===> CSR

Line Commands:                   Primary Commands:
   I - Interactive testing         Edit   - Display converted selected steps
   U - Unattended testing          END    - Exit without processing
   IC - Interactive Code Coverage  RUN    - Submit and connect
   UC - Unattended Code Coverage   SEtup  - Setup work datasets
                                   SUBmit - Convert selected steps and submit
   blank - Reset I/U/C             STatus - Display status of submitted job(s)

Dataset: 'XT.BATCON.JCL(BATCH)'

    PROGRAM     INITSCR      STEPNAME    PROCNAME    PROCSTEP     EXEC PGM
-------------------------   ----------------------------------------------
_   GENLABEL    ________     XMULTI      MULTI       STEP1        XPTSO
_   SORTMSTR    ________     XMULT1      MULT1       STEP2
_   PRINTCHK    ________     XMULT1      MULT1       STEP3        XPTSO
_   IDCAMS      ________     XMULT1      MULT1       STEP4
_   TRIIMSM     ________     XMULT1      MULT1       BATCHDLI
_   TRIBMP      ________     XMULT1      MULT1       BATCHIMS     XPBATCH
_   DSNMTV01    ________     XMULT1      MULT1       DB2DLI
_   PQ1CPINQ    ________     XMULT1      MULT1       CICSDLI
_               ________     XMULT1      MULT1       DSNRUN
_   XPTSO       ________     XMULT1      MULT1       XBATCONI
```

Figure 4-14. Select Job Step Screen

This screen is used two ways:

1. Use the I, IC, U, or UC line command to specify how you want each job step to execute: in interactive or unattended mode. The JCL will be converted to run with XPEDITER/TSO in the specified execution mode. Refer to “JCL Conversion” on page 4-24 for information about JCL conversion.
2. Use the RUN or SUBMIT primary command to convert and submit the JCL, the EDIT command to view or edit the JCL, the SETUP command to access and change your job card information, the STATUS command to display the status of submitted jobs, or the END command to exit.

For example, a completed screen is shown in Figure 4-15 on page 4-20.



```
------------------- XPEDITER/TSO - SELECT JOB STEP -------- ROW 00001 OF 00020
COMMAND ===>                                                 SCROLL ===> CSR

Line Commands:                    Primary Commands:
   I - Interactive testing          Edit   - Display converted selected steps
   U - Unattended testing           END    - Exit without processing
   IC - Interactive Code Coverage   RUN    - Submit and connect
   UC - Unattended Code Coverage    SEtup  - Setup work datasets
                                    SUBmit - Convert selected steps and submit
   blank - Reset I/U/C              STatus - Display status of submitted job(s)

Dataset: 'XT.BATCON.JCL(BATCH)'

    PROGRAM     INITSCR      STEPNAME    PROCNAME    PROCSTEP     EXEC PGM
-------------------------   --------------------------------------------
I   GENLABEL    ________     XMULTI      MULTI       STEP1        XPTSO
_   SORTMSTR    ________     XMULT1      MULT1       STEP2
I   PRINTCHK    ________     XMULT1      MULT1       STEP3        XPTSO
_   IDCAMS      ________     XMULT1      MULT1       STEP4
_   TRIIMSM     ________     XMULT1      MULT1       BATCHDLI
_   TRIBMP      ________     XMULT1      MULT1       BATCHIMS     XPBATCH
_   DSNMTV01    ________     XMULT1      MULT1       DB2DLI
_   PQ1CPINQ    ________     XMULT1      MULT1       CICSDLI
_               ________     XMULT1      MULT1       DSNRUN
_   XPTSO       ________     XMULT1      MULT1       XBATCONI
```

Figure 4-15. Completed Select Job Step Screen

Line Commands

I (Interactive) testing
Selects the step to run in interactive mode.

U (Unattended) testing
Selects the step to run in unattended mode.

IC (Interactive Code Coverage) testing
Selects the step to run in interactive mode (with Code Coverage active).

UC (Unattended Code Coverage) testing
Selects the step to run in unattended mode (with Code Coverage active).

blank (Reset I/U/C)
Blank the line command area and press **Enter** to remove an I, IC, U, or UC line command set on a job step.

Notes:

1. For a step that is already converted (EXEC PGM=XPTSO or EXEC PGM=XPBATCH), a question mark (?) is displayed and the step is not selectable.
2. If a job step cannot be resolved during JCL processing, the program name will be preceded by a question mark (?) and the I, IC, U, or UC line command cannot be used. See “JCL Conversion” on page 4-24 for additional information.

Primary Commands

Edit
Accesses the ISPF edit facility. The JCL is automatically converted and displayed on the ISPF edit screen. You can view and confirm the JCL conversion and make additional modifications to the converted JCL. However, the changes made to the converted JCL will not be saved when you CANCEL or PF3 (END) from the edit screen. Refer to “Editing the JCL” on page 4-22 for more information and an example of the ISPF edit screen.

You can submit the JCL from the edit screen using the RUN or SUBMIT command.

END
Exits without converting the JCL, saving any modifications, or submitting the job, and returns you to the Process JCL screen.

RUN
Converts the steps selected by the I and U line commands to XPEDITER/TSO steps and submits the JCL. When the job processes successfully, you are automatically connected to the job steps selected to run in interactive mode and the Source screen is displayed.

If the job is a long-running job, the Connect Status screen is displayed showing the job status.

Note: The terminal cannot be used while the job is running.

If the JCL is already converted, the RUN command (with a DSNAME) can be entered on any screen except an XPEDITER test session screen. See the *XPEDITER/TSO and XPEDITER/IMS Reference Manual* for more information about the RUN command.

SEtup
Displays the Setup Menu from which you can select to view and change the job card and library information, such as the DDIO file (XPSL000n) and the SCRIPT file (XINCLUDE). Refer to Appendix C, "Specifying Setup Options" for more information.

SUBmit
Converts the steps selected by the I, IC, U, and UC commands to XPEDITER/TSO steps and submits the job. A job submitted with the SUBMIT command **is not** automatically connected. To connect to a job submitted with the SUBMIT command, you must use the CONNECT or STATUS command. Refer to "Connecting to a Job" on page 4-21.

Note: While the job is running, you can continue using your terminal.

STatus
Displays the status of a job. You can connect to the job directly from the Status screen.

The fields displayed on the Select Job Step screen are:

Dataset
This field is pre-filled with the dataset name of the JCL being processed.

PROGRAM
The name of the program to be tested. The program name does not necessarily match the EXEC PGM name.

INITSCR
The member name of the script in the INCLUDE library specified on the Setup panel. The INITSCR field can be typed over to specify a test script member which can be processed at the beginning of a debugging session.

STEPNAME
The job step name.

PROCNAME
The in-stream or cataloged procedure name.

PROCSTEP
The step name within the called procedure.

EXEC PGM
The name of the EXEC program that is executed for the step. XPTSO if I (Interactive) or IC (Interactive Code Coverage) was specified for the step. XPBATCH if U (Unattended) or UC (Unattended Code Coverage) was selected for the step. The field is left blank if the name is the same as the one entered in the PROGRAM field.

Connecting to a Job

When you use the RUN command to submit the job, the steps that are selected for interactive debugging are automatically displayed at your terminal.

When you use the SUBMIT command to submit the job, there are two ways to connect to a job—the CONNECT command and the STATUS command.

CONNECT Command

Use the CONNECT command on any screen (except the source display) to connect a VTAM terminal to a job submitted through XPEDITER/TSO's batch connect facility. You can connect to a job with multiple steps or to a single step job. For information on the CONNECT command syntax, refer to the *XPEDITER/TSO and XPEDITER/IMS Reference Manual*.

STATUS command

The STATUS command is used to display the Status screen containing a list of the jobs in the system. The STATUS command can be entered from any screen.

The ATTACH line command on the Status screen is used to connect to a job and display the source of each job step for which the I (Interactive) command was specified. A message is displayed notifying you that the job step selected for testing is executing.

Connection Security Check

Connection can be made to any job, including production jobs, as long as your site security grants the authority. The batch connect facility is shipped with a default security exit routine that allows connection to a job if the JOBNAME, without the last character, matches the TSO ID where the STATUS panel is accessed. When a connection cannot be made, the messages `CANNOT CONNECT.......` or `SECURITY CHECK FAILED.......` are issued.

The site installer can customize the security exit routine to tailor the security level for certain groups or individuals. When an asterisk (*) is entered in the JOBNAME field on the Status screen, all jobs that are waiting for connection or being tested under batch connect are listed. System programmers are able to connect to a remote job and use the facility as a help desk feature in debugging application programs.

Editing the JCL

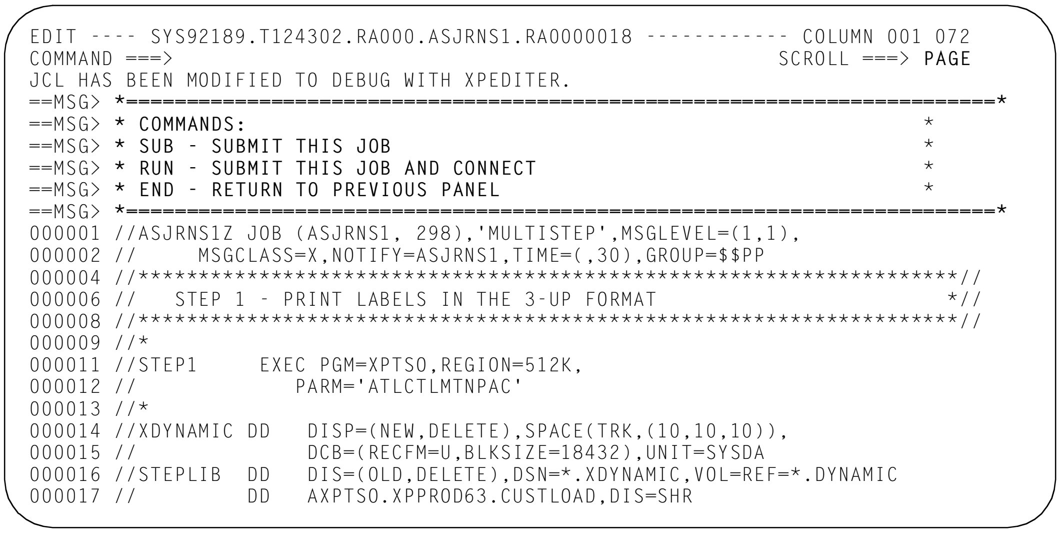
There are two points at which the batch connect facility lets you edit your JCL.

1. Primary editing is available by entering the EDIT primary command on the Process Execute JCL screen. An ISPF edit session is invoked and the specified JCL is displayed. If your site security permits, changes will be saved to the original JCL when the edit session ends.
2. Secondary editing is available by entering the EDIT primary command on the Select Job Step screen. An ISPF edit session is invoked and a temporary copy of the JCL is displayed. JCL statements for the steps selected for testing, using the I (Interactive) or U (Unattended) line commands, are already converted when the edit screen is displayed.

   After editing is complete, you can submit the job from this screen with the RUN or SUBMIT command.

   Note: Entering **END** (PF3) or **CANCEL** returns to the Select Job Step screen without saving the changes. To save the converted JCL, use the ISPF CREATE command and copy the contents to a dataset. The saved JCL can be submitted at any time with the RUN or SUBMIT command.

Figure 4-16 on page 4-23 is an example of the Edit screen. In this example, Step 1 is converted to run in interactive mode since the I line command was entered next to the job step on the Select Job Step screen.



```
EDIT ---- SYS92189.T124302.RA000.ASJRNS1.RA0000018 ------------ COLUMN 001 072
COMMAND ===>                                                  SCROLL ===> PAGE
JCL HAS BEEN MODIFIED TO DEBUG WITH XPEDITER.
==MSG> *===================================================================*
==MSG> * COMMANDS:                                                         *
==MSG> * SUB - SUBMIT THIS JOB                                             *
==MSG> * RUN - SUBMIT THIS JOB AND CONNECT                                 *
==MSG> * END - RETURN TO PREVIOUS PANEL                                    *
==MSG> *===================================================================*
//ASJRNS1Z JOB (ASJRNS1, 298),'MULTISTEP',MSGLEVEL=(1,1),
//     MSGCLASS=X,NOTIFY=ASJRNS1,TIME=(,30),GROUP=$$PP
//*******************************************************************//
//   STEP 1 - PRINT LABELS IN THE 3-UP FORMAT                        *//
//*******************************************************************//
//*
//STEP1     EXEC PGM=XPTSO,REGION=512K,
//             PARM='ATLCTLMTNPAC'
//*
//XDYNAMIC DD   DISP=(NEW,DELETE),SPACE(TRK,(10,10,10)),
//              DCB=(RECFM=U,BLKSIZE=18432),UNIT=SYSDA
//STEPLIB  DD   DIS=(OLD,DELETE),DSN=*.XDYNAMIC,VOL=REF=*.DYNAMIC
//         DD   AXPTSO.XPPROD63.CUSTLOAD,DIS=SHR
```

Figure 4-16. Edit Screen

Disconnecting the Terminal

Enter the **DISC** primary command from the Source display screen to disconnect the VTAM terminal from the batch job and return to the Status screen in TSO. The job can be reconnected by entering the A line command. A disconnect can also be accomplished by pressing the attention key twice.

General Information About Batch Connect

Testing Multiple Job Steps

The batch connect facility makes system testing easier for multiple step jobs by providing the capability of intercepting batch running applications and starting interactive debugging sessions. To understand how the facility works, consider the environment in which batch applications are executed.

Batch applications are processed by submitting jobs through JCL statements. A job can be simple or complex. It can consist of a single step or multiple steps that call in-stream and cataloged procedures. An example of a multiple step job is one that creates input control cards in the first step, processes the transaction file in the second step, and updates the master file in the third step.

In XPEDITER/TSO interactive mode, to debug a specific step in a job with multiple steps may require that you unit test the step by creating the necessary files and establishing the appropriate conditions to simulate that the previous steps were executed. You may also have to write a driver or stub routine to simulate the logic state that was set prior to reaching the step you want to test. This requires extensive planning and setup before debugging can begin.

Using the batch connect facility, you do not need to prepare a unit testing environment. When a job is submitted, steps that are selected for unattended testing execute normally. When a step is selected for interactive testing, XPEDITER/TSO gains control and connects the batch address space to a VTAM terminal. In effect, you are presented with an online debugging session that reads commands from and writes results to the screen while the program is actually executing in the batch region.

XPEDITER/TSO releases control when you resume execution by entering the GO command from the last logical statement in the program. Execution is passed to subsequent steps until an end-of-job is encountered.

JCL Conversion

Job steps marked with the I, IC, U, or UC line command on the Select Job Step screen are changed to XPEDITER/TSO steps in the JCL and generated to a temporary file. JCL conversion is as follows:

- XPEDITER/TSO work file DDs (XPSL000n, XPIN, XDYNAMIC, XINCLUDE, XOPTIONS, XPOUT, XPSHOW, XPHELP, and XPSCRIPT) are inserted.
- XPEDITER/TSO product library, XDYNAMIC, DDIO dataset, any libraries specified in environment setup, and the users application program are added to the STEPLIB. XPEDITER/TSO looks in the STEPLIB when attempting to intercept a program for debugging and analysis.

  If no STEPLIB exists, a STEPLIB is generated with the following concatenations:

  – XDYNAMIC

  – User libraries

  – XPEDITER/TSO product library

  – DDIO dataset

  – PCC (if installed)

  – JOBLIB (if a STEPLIB has not existed before)
- EXEC PGM= is updated as follows to point to the appropriate XPEDITER/TSO module name.

  – EXEC PGM=XPTSO (for interactive mode)

  – EXEC PGM=XPBATCH (for unattended mode)
- XPIN control cards are built. The control cards tell XPEDITER what environment and program to test.

If the JCL scanner did not parse the program name to be tested under XPEDITER, a question mark (?) is displayed in the column preceding the step on the Select Job Step screen. In this case, the I or U line command is not accepted and no conversion is made to allow interactive or unattended mode testing. Press the **PF1** key for an explanation. Following is a list of reasons why a step is not selectable:

1. **Step executes XPEDITER unattended batch**

   This step executes PGM=XPBATCH, the XPEDITER unattended batch program. A probable cause could be attempting to convert JCL that has already been coded to run XPEDITER unattended batch.
2. **Step executes XPEDITER interactive batch**

   This step executes PGM=XPTSO, the XPEDITER interactive batch program. A probable cause could be attempting to convert JCL that has already been coded to run XPEDITER interactive batch.
3. **PROC not found**

   The procedure cannot be found. A JCL error can occur if you submit this job.
4. **Stepname on an EXEC statement within a PROC must not be a blank**

   Unless it is the first step in a procedure, there must be a step name on the EXEC statement for a procedure step to be selected. Code a step name on the EXEC statement in the procedure definition and try again.
5. **No space in EXEC card for PGM substitution**

   The EXEC card did not have enough free space on it to allow conversion. Try splitting it into more than one card.
6. **PGM=DFHDRP is not supported**

This step executes PGM=DFHDRP, the CICS batch interface to DLI driver. JCL that uses the batch interface to DLI cannot be converted.

7. **PGM=BTSRC000 is not supported**

   This step executes PGM=BTSRC000, the IMS Batch Terminal Simulator (BTS). JCL that uses BTS cannot be converted.

   Note: You can access the editor and manually convert JCL that uses BTS by following the instructions in “Editing the JCL” on page 4-22.

Using XPEDITER/TSO Commands in Batch Connect Mode

Interactive Debugging

When you use the I (Interactive) line command to select a job step for interactive debugging, all XPEDITER/TSO COBOL commands except the following can be used.

```
ALLOC
FADB2
SET PFnn
```

You can also use script files in interactive mode.

- An initial script member name can be specified in the INITSCR field on the Select Job Step screen or you can edit your execution JCL and use the INITSCR parameter on the TEST or INTERCEPT command.
- A test script member name can be specified by using the INCLUDE command on the command line during the test session.
- A post script member name can be specified by editing your execution JCL and using the POSTSCR parameter on the TEST or INTERCEPT command.
- An abend script is not applicable in interactive mode.

For information about using commands in script files, refer to “Test Session Management Using Scripts” on page 4-42.

Unattended Debugging

When you use the U (Unattended) line command to select a job step for unattended testing, XPEDITER/TSO must read the predefined commands from a script file and write the results to the log file.

- An initial script member name can be specified in the INITSCR field on the Select Job Step screen or you can edit your execution JCL and use the INITSCR parameter on the TEST or INTERCEPT command.
- A test script member name can be specified by using the INCLUDE command in the job stream.
- A post script member name can be specified by editing your execution JCL and using the POSTSCR parameter on the TEST or INTERCEPT command.
- An abend script member name can be specified with the SET ABENDSCR command in the job stream.

For information about using scripts, refer to “Test Session Management Using Scripts” on page 4-42.

Intercepting Abends in Batch Connect Mode

Abend processing in a batch connect interactive session behaves the same as an XPEDITER/TSO interactive session. When an abend occurs, a message is displayed at the top of the screen informing you that you can use the AA SNAP command (if you have Abend-AID release 7.0.2 or above installed at your site) or the LOG command to view information about the abend. Please refer to Chapter 6, “Handling Run-Time Errors”.

In an unattended debugging session, the SET ABENDSCR command can be used in the job stream to specify the member containing the commands to be executed when an abend occurs. For example,in the following SET ABENDSCR command,

```
SET ABENDSCR TRIMABN1
```

TRIMABN1 is a member of a partitioned dataset allocated to the XINCLUDE DD.

The commands specified in an abend script are executed following the abend. This allows you to specify unique abend scripts containing special commands to be executed at different points in your code if an abend occurs.

For information about using abend scripts, refer to “Test Session Management Using Scripts” on page 4-42.

In an interactive or unattended session, the following information is written to the log when an abend occurs:

- Normal abend output.
- Source line where the abend occurred.
- A display for each field referenced in the source line where the abend occurred.

Usage Note

Even if your application programs would normally find any required Language dependent run-time subroutines (including LE - Language Environment), without being included in the JOBLIB/STEPLIB of the batch JCL (usually from the LINKLIST or (E)LPA), the libraries must still be specified as part of the test session setup. This will ensure that XPEDITER's Task Library will be properly configured. For Batch Connect, the preferred method is to include the run-time libraries in the STEPLIB DD statement concatenations of the JCL step(s) that are being intercepted.

Batch Connect Under ROSCOE

Starting A ROSCOE Batch Connect Session

With the batch connect facility, you can submit your execution JCL in batch and connect directly to the job as it executes in its native environment--the MVS batch initiator.

Notes:

1. In batch connect mode, you do not have to allocate the files and databases required for your program to run. They are automatically allocated to the batch address space via the DD statements in the JCL.
2. The size of programs which can be tested will not be limited by the size of the ROSCOE region.
3. The batch connect facility does not utilize ISPF services.
4. Batch connect does not support nested procedures.
5. The CICS interface to DLI driver (DFHDRP) is not supported in batch connect.

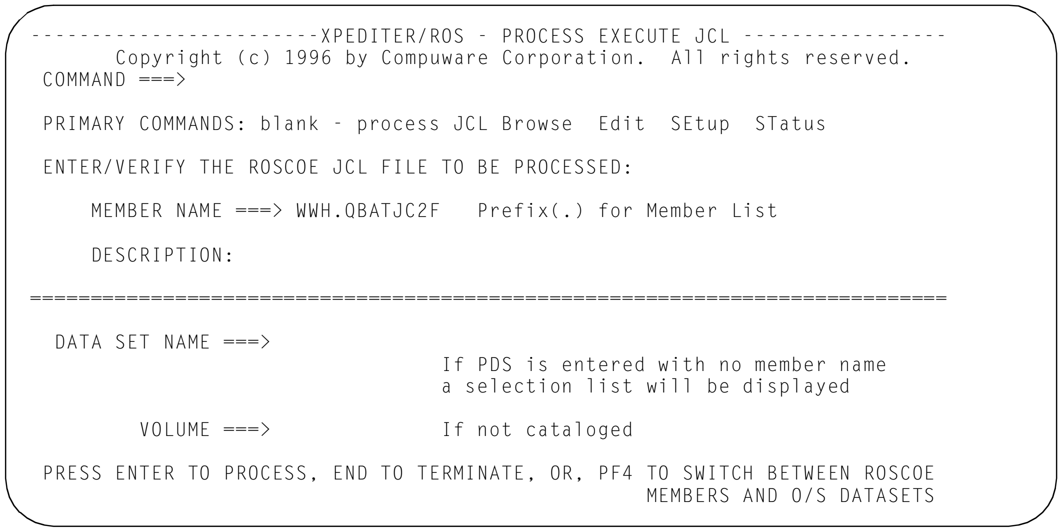
The batch connect facility presents several screens that are used to automate the process of starting the session. The following summarizes the process:

1. ACCESS THE FACILITY: From the ROSCOE command line enter the command XBATCON. The Process Execute JCL screen is displayed.
2. PROCESS THE EXECUTION JCL: The Process Execute JCL screen is used to identify the execution JCL for the debugging session. You can also use ROSCOE services to view and make changes to the JCL. The JCL statements are scanned, expanded if the JCL is a procedure, and the job steps are extracted. A list of the job steps is then displayed on the Select Job Step screen.

3. CONVERT AND SUBMIT THE JCL: The Select Job Step screen is used to specify how you want each job step to execute (in interactive or unattended mode). When a job step is selected for interactive debugging, the source for that step is displayed at your terminal. When a step is selected to run in unattended mode, you cannot interact with the step from your terminal. If you want to debug the step, XPEDITER debugging commands must be read from a test script and the output results from the test session are written to the log file. You can submit the job from this screen or you can edit the JCL. When you submit the job, the batch connect intelligent scanner automatically converts each step to execute in the specified mode (interactive or unattended).
4. CONNECT TO THE JOB: If you submit the job with the PROCESS command, your job is submitted and you are taken directly to the STATUS screen with your job being displayed. If you use the SUBMIT command, you must use the STATUS command to connect your terminal to the job steps selected for interactive debugging.
5. EDIT THE JCL: Optionally, you can access the edit facility from the Process Execute JCL or Select Job Step screen to view and confirm the JCL conversion or to make changes to the JCL.
6. ACCESS THE SETUP FACILITY: Optionally, you can use the SETUP command on the Process Execute JCL screen or the Select Job Step screen to access the Setup Menu. From the Setup Menu you can view and change the installed defaults such as the load libraries, DDIO dataset, etc.

Processing The JCL

The Process Execute JCL screen shown below is used to specify the name of the JCL to be used for the debugging session and to specify what you want to do with the JCL:



```
-----------------------XPEDITER/ROS - PROCESS EXECUTE JCL ----------------
      Copyright (c) 1996 by Compuware Corporation.  All rights reserved.
 COMMAND ===>

 PRIMARY COMMANDS: blank - process JCL Browse  Edit  SEtup  STatus

 ENTER/VERIFY THE ROSCOE JCL FILE TO BE PROCESSED:

     MEMBER NAME ===> WWH.QBATJC2F   Prefix(.) for Member List

     DESCRIPTION:

============================================================================

  DATA SET NAME ===>
                                 If PDS is entered with no member name
                                 a selection list will be displayed

        VOLUME ===>              If not cataloged

 PRESS ENTER TO PROCESS, END TO TERMINATE, OR, PF4 TO SWITCH BETWEEN ROSCOE
                                                  MEMBERS AND O/S DATASETS
```

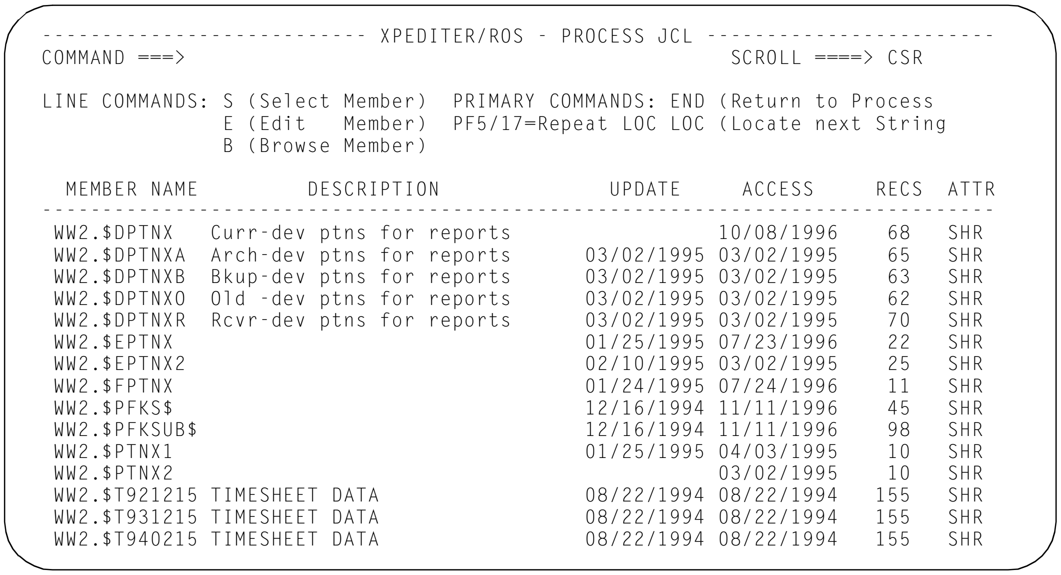
Figure 4-17. Process Execute JCL For XPEDITER/ROSCOE

There are two things you need to do on this screen:

1. Enter the name of the dataset or ROSCOE member that contains the execution JCL for the program(s) you wish to debug. ROSCOE library members are entered in the top half of the panel and O/S datasets are entered in the bottom half. If a ROSCOE user prefix is entered with a period (ABC.), a ROSCOE library selection list will be shown. If an O/S PDS dataset is entered without a member name, a PDS selection list will be displayed. Pressing the PF4 key will switch between ROSCOE and O/S resources.
2. Use one of the following commands on the command line:

BLANK	Leave the command line blank to scan the specified JCL and extract the job step information. If the JCL is a procedure, it is expanded prior to scanning.
BROWSE	Invokes a controlled ROSCOE browse facility and displays the specified JCL.
EDIT	Invokes a controlled ROSCOE edit facility and displays the specified JCL. You can edit the JCL.
SETUP	Accesses the Test Setup Menu which will allow you to specify XPEDITER and user libraries that are to be used.
STATUS	Displays the status of any submitted job(s). You can connect to a job from the Status screen.

Member Selection List Processing



```
--------------------------- XPEDITER/ROS - PROCESS JCL ------------------------
COMMAND ===>                                              SCROLL ====> CSR

LINE COMMANDS: S (Select Member)  PRIMARY COMMANDS: END (Return to Process
               E (Edit   Member)  PF5/17=Repeat LOC LOC (Locate next String
               B (Browse Member)

  MEMBER NAME         DESCRIPTION              UPDATE     ACCESS     RECS  ATTR
--------------------------------------------------------------------------------
 WW2.$DPTNX   Curr-dev ptns for reports                 10/08/1996    68   SHR
 WW2.$DPTNXA  Arch-dev ptns for reports      03/02/1995 03/02/1995    65   SHR
 WW2.$DPTNXB  Bkup-dev ptns for reports      03/02/1995 03/02/1995    63   SHR
 WW2.$DPTNXO  Old -dev ptns for reports      03/02/1995 03/02/1995    62   SHR
 WW2.$DPTNXR  Rcvr-dev ptns for reports      03/02/1995 03/02/1995    70   SHR
 WW2.$EPTNX                                  01/25/1995 07/23/1996    22   SHR
 WW2.$EPTNX2                                 02/10/1995 03/02/1995    25   SHR
 WW2.$FPTNX                                  01/24/1995 07/24/1996    11   SHR
 WW2.$PFKS$                                  12/16/1994 11/11/1996    45   SHR
 WW2.$PFKSUB$                                12/16/1994 11/11/1996    98   SHR
 WW2.$PTNX1                                  01/25/1995 04/03/1995    10   SHR
 WW2.$PTNX2                                             03/02/1995    10   SHR
 WW2.$T921215 TIMESHEET DATA                 08/22/1994 08/22/1994   155   SHR
 WW2.$T931215 TIMESHEET DATA                 08/22/1994 08/22/1994   155   SHR
 WW2.$T940215 TIMESHEET DATA                 08/22/1994 08/22/1994   155   SHR
```

Figure 4-18. Process JCL Screen for XPEDITER/ROSCOE

This panel is displayed if a ROSCOE prefix was entered in the previous panel (E.G. WW2.) or if a PDS dataset was entered without a member that had previously been specified.

One of the following actions must be taken:

1. Select a member for processing by typing an **S** in front of it.
2. Select a member to Browse by typing a **B** in front of it.
3. Select a member to Edit by typing an **E** in front of it.
4. Return to the previous panel by pressing the PF3/PF15 keys.

In addition, the selection list can be searched by using the LOC primary command followed by the search argument. The command LOC JCL will reposition the display to start with the first line that contains the string JCL. Pressing the PF5 key will repeat the LOC command.

Establishing The Batch Connect Environment

The SEtup panels allow the user to customize the Batch Connect environment. Additional STEPLIB datasets can be specified and the XPEDITER required datasets can be customized. As part of the XPEDITER install, system defaults will be initialized for each of these catagories. Each user can further customize the environment by use of this setup facility. Except for the Log and Script files, selecting the desired option will display a screen that allows modification to both user overrides and the system defaults (for the specific user only). The PROCLIBS option allows the user to specify an additional datatset that XPEDITER should use in resolving procedures.

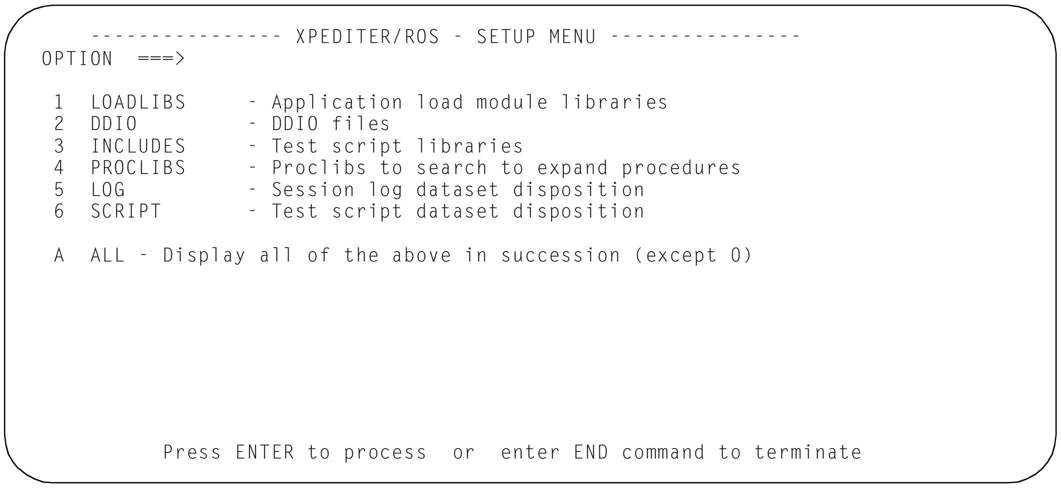

```
    --------------- XPEDITER/ROS - SETUP MENU ---------------
OPTION  ===>

 1  LOADLIBS      - Application load module libraries
 2  DDIO          - DDIO files
 3  INCLUDES      - Test script libraries
 4  PROCLIBS      - Proclibs to search to expand procedures
 5  LOG           - Session log dataset disposition
 6  SCRIPT        - Test script dataset disposition

 A  ALL - Display all of the above in succession (except 0)




          Press ENTER to process  or  enter END command to terminate
```

Figure 4-19. Setup Menu Screen For XPEDITER/ROSCOE

Converting And Submitting The JCL

When JCL processing is completed, the job steps in the specified JCL are displayed on the Select Job Step screen as shown below:

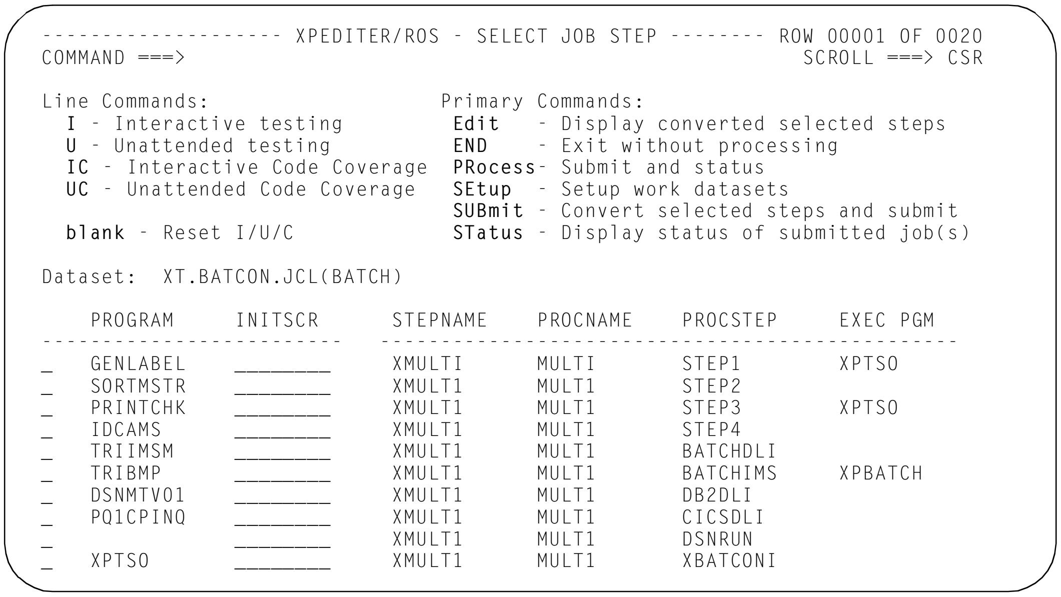
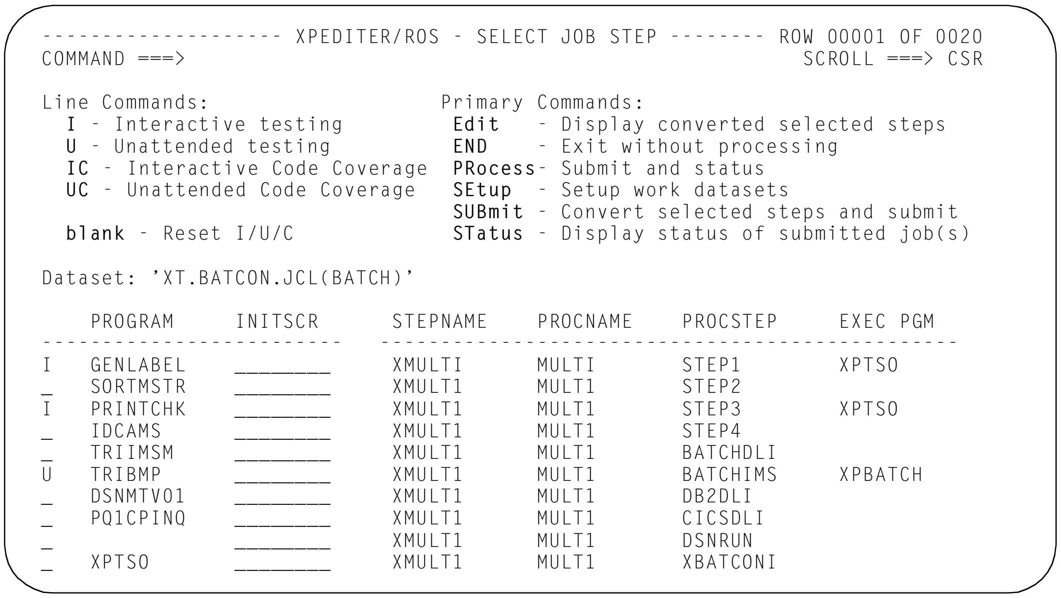

```
    ------------------- XPEDITER/ROS - SELECT JOB STEP -------- ROW 00001 OF 0020
COMMAND ===>                                                  SCROLL ===> CSR

Line Commands:                    Primary Commands:
  I - Interactive testing          Edit    - Display converted selected steps
  U - Unattended testing           END     - Exit without processing
  IC - Interactive Code Coverage   PRocess- Submit and status
  UC - Unattended Code Coverage    SEtup   - Setup work datasets
                                   SUBmit  - Convert selected steps and submit
  blank - Reset I/U/C              STatus  - Display status of submitted job(s)

Dataset:  XT.BATCON.JCL(BATCH)

    PROGRAM     INITSCR      STEPNAME    PROCNAME    PROCSTEP     EXEC PGM
-------------------------   -----------------------------------------------
_   GENLABEL    ________     XMULTI      MULTI       STEP1        XPTSO
_   SORTMSTR    ________     XMULT1      MULT1       STEP2
_   PRINTCHK    ________     XMULT1      MULT1       STEP3        XPTSO
_   IDCAMS      ________     XMULT1      MULT1       STEP4
_   TRIIMSM     ________     XMULT1      MULT1       BATCHDLI
_   TRIBMP      ________     XMULT1      MULT1       BATCHIMS     XPBATCH
_   DSNMTV01    ________     XMULT1      MULT1       DB2DLI
_   PQ1CPINQ    ________     XMULT1      MULT1       CICSDLI
_               ________     XMULT1      MULT1       DSNRUN
_   XPTSO       ________     XMULT1      MULT1       XBATCONI
```

Figure 4-20. Select Job Step Screen For XPEDITER/ROSCOE

This screen is used two ways:

1. Use the I, U, IC, or UC line commands to specify how you want each job step to execute, in interactive or unattended mode. The JCL will be converted to run with XPEDITER/ROSCOE in the specified execution mode.
2. Use the PROcess or SUBmit primary command to convert and submit the JCL, the EDIT command to view or edit the JCL, the SETUP command to access and change XPEDITER/ROSCOE Batch Connect environment, the STATUS command to display the status of submitted jobs, or the END command to exit.

For example, a completed screen is shown in the figure below:



```
------------------- XPEDITER/ROS - SELECT JOB STEP -------- ROW 00001 OF 0020
COMMAND ===>                                                 SCROLL ===> CSR

Line Commands:                   Primary Commands:
  I - Interactive testing         Edit    - Display converted selected steps
  U - Unattended testing          END     - Exit without processing
  IC - Interactive Code Coverage  PRocess- Submit and status
  UC - Unattended Code Coverage   SEtup   - Setup work datasets
                                  SUBmit - Convert selected steps and submit
  blank - Reset I/U/C             STatus - Display status of submitted job(s)

Dataset: 'XT.BATCON.JCL(BATCH)'

    PROGRAM     INITSCR      STEPNAME    PROCNAME    PROCSTEP     EXEC PGM
-------------------------   --------------------------------------------------
I   GENLABEL    ________     XMULTI      MULTI       STEP1        XPTSO
_   SORTMSTR    ________     XMULT1      MULT1       STEP2
I   PRINTCHK    ________     XMULT1      MULT1       STEP3        XPTSO
_   IDCAMS      ________     XMULT1      MULT1       STEP4
_   TRIIMSM     ________     XMULT1      MULT1       BATCHDLI
U   TRIBMP      ________     XMULT1      MULT1       BATCHIMS     XPBATCH
_   DSNMTV01    ________     XMULT1      MULT1       DB2DLI
_   PQ1CPINQ    ________     XMULT1      MULT1       CICSDLI
_               ________     XMULT1      MULT1       DSNRUN
_   XPTSO       ________     XMULT1      MULT1       XBATCONI
```

Figure 4-21. Completed Select Job Step Screen For XPEDITER/ROSCOE

Line Commands

I (INTERACTIVE)
Selects the step to run in interactive mode.

U (UNATTENDED)
Selects the step to run in unattended mode.

IC (INTERACTIVE CODE COVERAGE)
Selects the step to run in interactive mode (with Code Coverage active).

UC (UNATTENDED CODE COVERAGE)
Selects the step to run in unattended mode (with Code Coverage active).

BLANK (RESET I/U/C)
Blank the line command area and press **Enter** to remove an I, I/C, U, or U/C line command set on a job step.

Note: If a job step cannot be resolved during JCL processing, the program name will be preceded by a question mark (?) and the I or U will be preceded by a question mark (?).

Primary Commands

EDIT Accesses a controlled RPF edit facility. The JCL is automatically converted and displayed on the controlled RPF edit screen. You can view and confirm the JCL conversion and make additional modifications to the converted JCL. However, the changes made to the converted JCL **WILL NOT** be saved when you CANCEL or PF3 (END) from the edit screen.

You can submit the JCL from the edit screen using the SUBMIT command.

END Exits without converting the JCL, saving any modifications, or submitting the job and returns you to the Process JCL screen.

PROCESS Converts the steps selected by the I and U line commands to XPEDITER/ROSCOE steps and submits the JCL. It then automatically displays the status screen to allow monitoring the job's process.

SETUP Displays the Setup Menu from which you can select to view and change the job card information and library information, such as the DDIO file (XPSL000n) and the SCRIPT file (XINCLUDE).

SUBMIT Converts the steps selected by the I, IC, U, and UC commands to XPEDITER/ROSCOE steps and submits the job. A job submitted with the SUBMIT command **IS NOT** automatically connected. To connect to a job submitted with the SUBMIT command, you must use the CONNECT or STATUS command.

Note: While the job is running, you can continue using your terminal.

STATUS Displays the status of a job. You can connect to the job from the Status screen.

The fields on the Select Job Step screen are:

DATASET	This field is prefilled with the dataset name of the JCL being processed.
PROGRAM	The name of the program to be tested. The program name does not necessarily match the EXEC PGM name.
INITSCR	The member name of the script in the INCLUDE library specified on the Setup panel. The INITSCR field can be typed over to specify a test script member which can be processed at the beginning of a debugging session.
STEPNAME	The job step name.
PROCNAME	The in-stream or cataloged procedure name.
PROCSTEP	The step name within the called procedure.
EXEC PGM	The name of the EXEC program that is executed for the step. XPTSO if I (Interactive) or IC (Interactive Code Coverage) was specified for the step. XPBATCH if U (Unattended) or UC (Unattended Code Coverage) was selected for the step. The field is left blank if the name is the same as the one entered in the PROGRAM field.

Connecting To A Job

When you use the PROcess command to submit the job, the job is submitted and the status screen is displayed.

When you use the SUBMIT command to submit the job, the STATUS command must then be used to connect to the job.

STATUS command: The STATUS command is used to display the Status screen containing a list of the jobs in the system.

The ATTACH line command on the Status screen is used to connect to a job and display the source of each job step for which the I (Interactive) command was specified.

Although no message is generated when a job is ready to attach, it is safe to assume that when a job is no longer using CPU time, that job is ready to attach.

Connection Security Check

Connection can be made to any job, including production jobs, as long as your site security grants the authority. The batch connect facility is shipped with a default security exit routine that allows connection to a job if the JOBNAME, without the last character, matches the TSO ID where the STATUS panel is accessed. When a connection cannot be made, the message "**CANNOT CONNECT.......**" is issued.

The site installer can customize the security exit routine to tailor the security level for certain groups or individuals.

Editing The JCL

There are two points at which the batch connect facility allows you to edit your JCL.

1. Primary editing is available by entering the EDIT primary command on the Process Execute JCL screen. A controlled RPF edit facility session is invoked and the specified JCL is displayed. If your site security permits, changes will be saved to the original JCL when the edit session ends.
2. Secondary editing is available by entering the EDIT primary command on the Select Job Step screen. A controlled RPF edit facility session is invoked and a temporary copy of the JCL is displayed. JCL statements for the steps selected for testing, using the I (Interactive) or U (Unattended) line commands, are already converted when the edit screen is displayed.

After editing is complete, you can submit the job from this screen with the RUN or SUBMIT command.

Note: Entering END (PF3) or CANCEL returns to the Select Job Step screen without saving the changes. To save the converted JCL, use the normal ROSCOE SAVE, UPDATE, or EXPORT commands. The saved JCL can be submitted at any time with the SUBMIT command.

General Information About Batch Connect

Testing Multiple Job Steps

The batch connect facility makes system testing easier for multiple step jobs by providing the capability of intercepting batch running applications and starting interactive debugging sessions. To understand how the facility works, consider the environment in which batch applications are executed.

Batch applications are processed by submitting jobs through JCL statements. A job can be simple or complex. It can consist of a single step or multiple steps that call in-stream and cataloged procedures. An example of a multiple step job is one that creates input control cards in the first step, processes the transaction file in the second step, and updates the master file in the third step.

In XPEDITER/ROSCOE interactive mode, to debug a specific step in a job with multiple steps may require that you unit test the step by creating the necessary files and establishing the appropriate conditions to simulate that the previous steps were executed. You may also have to write a driver or stub routine to simulate the logic state that was set prior to reaching the step you want to test. This requires extensive planning and setup before debugging can begin.

Using the batch connect facility, you do not need to prepare a unit testing environment. When a job is submitted, steps that are selected for unattended testing execute normally. When a step is selected for interactive testing, XPEDITER/ROSCOE gains control and connects the batch address space to a VTAM terminal. In effect, you are presented with an online debugging session that reads commands from and writes results to the screen while the program is actually executing in the batch region.

XPEDITER/ROSCOE releases control when you resume execution by entering the GO command from the last logical statement in the program. Execution is passed to subsequent steps until an end-of-job is encountered.

JCL Conversion

Job steps marked with the I, IC, U, or UC line command on the Select Job Step screen are changed to XPEDITER/ROSCOE steps in the JCL and generated to a temporary file. JCL conversion is as follows:

1. XPEDITER/ROSCOE work file DDs (XPSL000n, XPIN, XDYNAMIC, XINCLUDE, XPOUT, XPSHOW, XPHELP, XPSCRIPT) are inserted.
2. XPEDITER/ROSCOE product library, XDYNAMIC, DDIO dataset, any libraries specified in environment setup, and the users application program are added to the

STEPLIB. XPEDITER/ROSCOE looks in the STEPLIB when attempting to intercept a program for debugging and analysis.

3. If no STEPLIB exists, a STEPLIB is generated with the following concatenations:
   - XDYNAMIC
   - User Libraries
   - XPEDITER/ROSCOE Product Library
   - DDIO dataset
   - JOBLIB (if a STEPLIB has not existed before)
4. EXEC PGM= is updated as follows to point to the appropriate XPEDITER/ROSCOE module name.
   - EXEC PGM=XPTSO (for interactive mode)
   - EXEC PGM=XPBATCH (for unattended mode)
5. XPIN control cards are built. The control cards tell XPEDITER what environment and program to test.

Using XPEDITER/ROSCOE Commands in Batch Connect Mode

Interactive Debugging

When you use the I (INTERACTIVE) line command to select a job step for interactive debugging, all XPEDITER/ROSCOE commands except the following can be used:

```
ALLOC
SET PFnn
```

You can also use script files in interactive mode. However, keep in mind that only primary commands can be used in a script file.

- An initial script member name can be specified in the INITSCR field on the Select Job Step screen or you can edit your execution JCL and use the INITSCR parameter on the TEST or INTERCEPT commands.
- A test script member name can be specified by using the INCLUDE command on the command line during the test session.
- A post script member name can be specified by editing your execution JCL and using the POSTSCR parameter on the TEST or INTERCEPT commands.
- An abend script is not applicable in interactive mode.

Intercepting Abends in Batch Connect Mode

Abend processing in a batch connect interactive session behaves the same as an XPEDITER/ROSCOE interactive session. When an abend occurs, a message is displayed at the top of the screen informing you that you can use the AA SNAP command (if you have Abend-AID release 7.0.2 or above installed at your site) or the LOG command to view information about the abend.

Test Session Management Using Scripts

A script is a predefined stream of XPEDITER/ROSCOE commands that can be used to:

1. Initialize a debugging session (initial script).
2. Execute a set of commands during the debugging session (test script)
3. Execute a set of commands at the end of a session (post script)
4. Execute a set of commands when an abend occurs in an unattended batch session (abend script).

Scripts enable you to:

- Eliminate redundant keystrokes.
- Play back the commands established during a previous session.
- Run regression testing.

You create and maintain the scripts and the script libraries you use. A script library is a partitioned dataset (PDS) that can be FB 80 (fixed block with a record length of 80) or a VB 255 (variable blocked with a maximum record length of 255). If you use FB 80, only the data in columns 1 through 72 is recognized.

If you intend to use a script during the test session, the script library must be preallocated before the session begins. Use the SETUP command and select Option 3 (INCLUDES) on the Setup Menu to allocate the library, unless the script member is contained in a a Site-wide library specified at installation time.

Initial Script

An initial script is executed at the beginning of the debugging session or at the BEFORE breakpoint of each module. It can be used to:

- Set initial PF key values.
- Specify XPEDITER/ROSCOE debugging session processing options with the SET command.
- Set initial breakpoints in modules.

In interactive mode, the initial script member name is specified in the Initial Script field on the selected environment test screen.

In batch connect interactive or unattended mode, the initial script member name is specified in the INITSCR field on the Select Job Step screen or you can edit your execution JCL and use the INITSCR parameter on the TEST or INTERCEPT commands.

The ability to process SET commands before the program is loaded is the most useful feature of the initial script. The following SET commands either configure XPEDITER/ROSCOE or control the manner in which XPEDITER/ROSCOE loads programs:

```
SET DYNAMIC            SET STATIC
SET EXCLUDE            SET TRANSFER
```

The following commands determine the size of the log and the way information is represented.

```
SET HEXMODE
SET LOGSIZE
SET NONDISP
```

Test Script

A test script is used to execute XPEDITER/ROSCOE debugging commands during a debugging session. The commands in the test script are executed in the order they are read, as if they had been entered serially from the terminal.

In interactive mode or batch connect interactive, use the INCLUDE command with the test script member name at any time within the session to execute a script command stream.

In unattended batch, the test script member name is specified by using the INCLUDE command in the JCL job stream.

In interactive mode, XPEDITER/ROSCOE automatically generates a test script of all the commands entered during the debugging session. The generated file can be edited and copied into a script library for later use when you want to duplicate the debugging session. The generated test script is accessed by typing the SCRIPT command on the command line of the Test screen after a debugging session.

Post Scripts

A post script is comprised of a command or set of commands that are executed when the end of the program is encountered. The commands are effectively executed after the debugging session is ended, but before the XPEDITER/ROSCOE environment is exited.

A post script has many purposes. For instance, it lets you display (PEEK) the value of variables at the close of the debugging session and show the count tallied on program statements.

In interactive mode, a post script member name is specified in the Post Script field on the selected environment test screen.

In batch connect interactive or unattended mode, a post script member name is specified by editing your execution JCL and using the POSTSCR parameter on the TEST or INTERCEPT commands.

Note: It is suggested that you use PEEK instead of KEEP in a post script.

Abend Scripts

An abend script is an XPEDITER/ROSCOE command or set of commands that are executed when an abend occurs. You can use an abend script only when debugging in unattended mode.

Use the SET ABENDSCR command in the JCL job stream to specify the abend script to be executed when an abend occurs. The commands included in that script are executed whenever an abend occurs, or until another SET ABENDSCR command is executed. Any number of SET ABENDSCR commands can be included in a job stream.

Note: It is suggested that you use PEEK instead of KEEP in an abend script.

Creating And Editing Scripts

In a script, the following must be observed:

1. Commands cannot exceed 61 characters.
2. Commands must be entered in uppercase. Lowercase characters are not valid.
3. Only one XPEDITER/ROSCOE command on a single line.
4. A command can be continued beyond a single line without a continuation character.
5. A quoted string must be contained on one line.
6. Test scripts can be nested without limit by inserting additional INCLUDE commands within the test script.
7. Comment lines can be included by entering an asterisk (*) in column 1.
8. When used within inserted code, the INCLUDE command will be executed when the inserted code is executed.

Scripts are extremely useful under certain debugging circumstances. For example, suppose you end a debugging session knowing that at some later time you intend to retest the program along the same lines. A script is an efficient way to quickly reproduce that session without having to reenter the commands individually.

Saving And Using Generated Scripts

In interactive mode, XPEDITER/ROSCOE automatically generates a test script of all the commands entered during the debugging session. The following steps will show you how to save and use the script for later use when you want to duplicate the debugging session:

1. When the debugging session is completed, the message "Log and Script Created" appears in the upper right corner of the Test screen. Enter SCRIPT on the command line of the Test screen.
2. Press PF3 to save the edited script.
3. On the Data Set Disposition Screen, enter **C** in the Process Option field to copy the edited script to a partitioned dataset (PDS).
4. If you have already allocated a PDS for this purpose, enter the PDS name in the DSNAME field of this screen and a name for the edited script in the Member Name field. Press ENTER and skip to Step 6. Otherwise, go to the next step.
5. If you have not allocated a PDS, you can do so at this time by entering a library name in the DSNAME field of the screen and a name for the edited script in the Member Name field. Press ENTER and the New Dataset allocation screen is displayed with the dataset name you selected, prefilled in the Dataset Name field. Enter the parameters for the dataset according to your site standards and press ENTER to process.
6. You are returned to the Test screen and a message specifying the number of lines that were copied is displayed in the upper right corner of the screen. If your PDS is not specified in setup on the Test Script Libraries screen, enter SETUP from the Test screen and enter 3 on the Setup Menu screen. Then enter the PDS name on the Test Script Libraries screen. Press ENTER to process.
7. Press PF3 to return to the Test screen.
8. When you want to use the script for a subsequent test session, specify the PDS member name of the script in the Test screen’s Initial Script field or use the INCLUDE command followed by the PDS member name to execute the script after the source is displayed.

Note: The LOG and SCRIPT commands are still displayed on the Test screen. This means that the log and script files from the previous session are still active. If you intend to start another debugging session and want to use fresh log and script files, press PF3 to return to the XPEDITER/ROSCOE Primary Menu and then enter option 2 to return to the Test screen. This deletes the previous log and script files and you are ready to begin a new session.

Script Example

Suppose you want to set a number of breakpoints that are sufficiently spaced apart and require you to scroll up after setting each breakpoint. A script comprised of these breakpoint commands could initialize the session, setting the breakpoints. This would eliminate the need to scroll in order to enter the commands.

```
BEFORE 24  30  TRIMAINP:
AFTER TRITSTP:DETERMINE_TYPE
COUNT  24  30
```

Using XPEDITER/ROSCOE Commands in a Script (Interactive)

The following commands can be included in a script for interactive debugging of Assembler or PL/I programs:

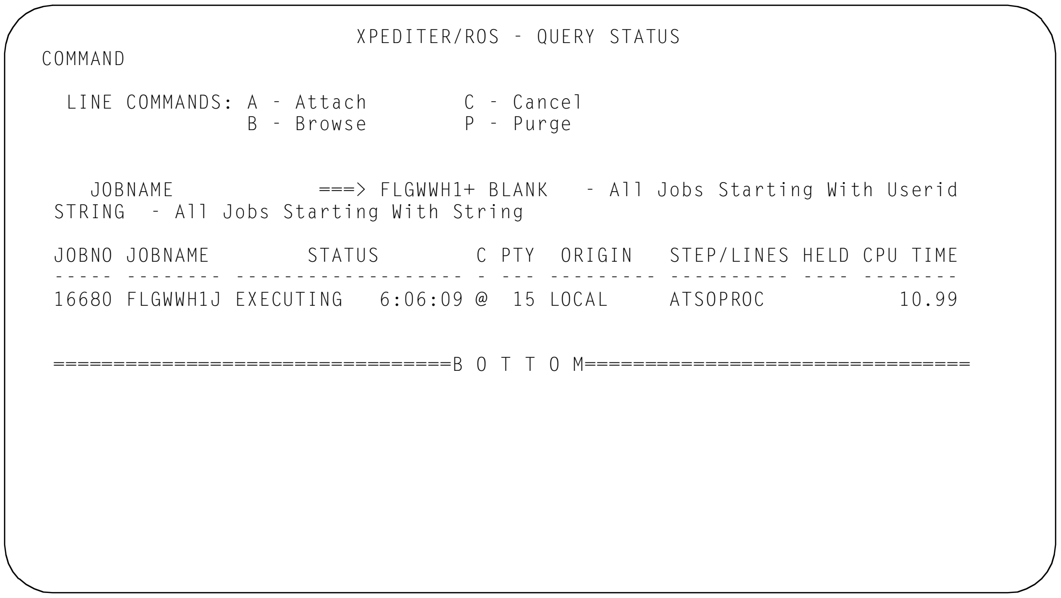
```
AA SNAP      DELETE      IF           MOVE       SHOW
AFTER        EXCLUDE     INCLUDE      PAUSE      SKIP
AT           EXIT        INTERCEPT    PEEK       SOURCE
BEFORE       GO          KEEP         RESET      TRACE
BROWSE       GOTO        LINE         RETURN     WHEN
COUNT        GPREGS      LOAD         SET
```

Note: The PAUSE and IF commands can be used only within an insert.

The following commands cannot be included in a script for interactive debugging of Assembler or PL/I programs:

```
        ALLOC       DRIGHT      LOCATE      RUN         WHEREIS
        BOTTOM      END         LOG         STATUS      XCHANGE
        CONNECT     FADB2       MEMORY      TEST        XPED
        DLEFT       FIND        NOLINES     TOP
        DLI         HELP        RETEST      TSO
        DOWN        LEFT        RIGHT       UP
-------------------------------------------------------------------
---
```

Displaying And Attaching To A Batch Connect Job



```
                           XPEDITER/ROS - QUERY STATUS
COMMAND

  LINE COMMANDS: A - Attach        C - Cancel
                 B - Browse        P - Purge


    JOBNAME           ===> FLGWWH1+ BLANK   - All Jobs Starting With Userid
 STRING  - All Jobs Starting With String

 JOBNO JOBNAME        STATUS        C PTY  ORIGIN   STEP/LINES HELD CPU TIME
 ----- -------- ------------------- - --- --------- ---------- ---- --------
 16680 FLGWWH1J EXECUTING   6:06:09 @  15 LOCAL     ATSOPROC            10.99


 =================================B O T T O M================================
```

Figure 4-22. Query Status Screen For XPEDITER/ROSCOE

Starting a Session With Batch JCL

This section discusses how to manually change your execution JCL to invoke an XPEDITER/TSO debugging session in unattended or interactive (batch connect) batch modes. No file allocation utility is required and you can even use your production JCL for IMS/DB, BTS, and DB2 programs.

To debug your program with XPEDITER/TSO in unattended or interactive batch, follow the steps listed below:

1. Retrieve your standard TSO program execution JCL.
2. Follow the instructions in member BATCHTSO from the SAMPLIB or SORCMAC dataset and make the appropriate changes to the JCL.
3. Submit the job. If you are running unattended batch, the test results will be saved in the XPOUT file. If you are using interactive batch, you must connect to the job to complete testing.

Figure 4-23 shows JCL that runs a simple job without XPEDITER/TSO. Figure 4-24 on page 4-44 is an example of the JCL after it is modified to run with XPEDITER/TSO. Refer to "Setting Up the Batch JCL" on page 4-45 for an explanation of the modifications.

```
//MYJOB    JOB (ACCOUNTING),
//*            NOTIFY=TSOUSER,
//             CLASS=A,MSGCLASS=A,
//             MSGLEVEL=(1,1),TIME=(,10)
//*
//*  RUN TRIMAIN IN BATCH WITHOUT XPEDITER/TSO
//*
//MYTEST   EXEC PGM=TRIMAIN
//STEPLIB  DD  DSN=USER.LOADLIB,DISP=SHR
//*
//*   ALLOCATE ALL INPUT AND OUTPUT DDNAMES
//*
//INFILE   DD  DSN=SYS2.XPEDITER.V7R0M0.SAMPLIB(TRIDATA),
//             DISP=SHR
//OUTFILE  DD  SYSOUT=(*)
//SYSOUT   DD  SYSOUT=(*)
```

Figure 4-23. JCL to Run a Program Without XPEDITER/TSO

```
//MYJOB    JOB (ACCOUNTING),
//*            NOTIFY=TSOUSER,                                      NOTE 1
//             CLASS=A,MSGCLASS=A,                                  NOTE 2
//             MSGLEVEL=(1,1),TIME=(,10)
//*
//*  RUN TRIMAIN IN INTERACTIVE BATCH WITH XPEDITER/TSO
//*
//CREATE   EXEC PGM=IEFBR14                                          NOTE 3
//XPLOG    DD  DSN=&&XPLOG,
//             DISP=(NEW,PASS),UNIT=SYSDA,
//             SPACE=(TRK,(2,2))
//MYTEST   EXEC PGM=XPTSO,REGION=4098K                               NOTE 4
//XDYNAMIC DD  DISP=(NEW,DELETE),                                    NOTE 5
//             UNIT=SYSDA,
//             DCB=(RECFM=U,BLKSIZE=BBBBB,DSORG=PO),
//             SPACE=(CYL,(5,1,10))
//STEPLIB  DD  DISP=(OLD,DELETE),DSN=*.XDYNAMIC,VOL=REF=*.XDYNAMIC   NOTE 6
//         DD  DSN=SYS2.XPEDITER.V7R0M0.LOADLIB,DISP=SHR
//         DD  DSN=COMPWARE.CSS.LOADLIB,DISP=SHR
//         DD  DSN=YOUR.USER.LOADLIB,DISP=SHR                        NOTE 7
//*
//*   ALLOCATE ALL INPUT AND OUTPUT DDNAMES
//*
//INFILE   DD  DSN=SYS2.XPEDITER.V7R0M0.SAMPLIB (TRIDATA),           NOTE 8
//             DISP=SHR
//OUTFILE  DD  SYSOUT=(*)
//SYSOUT   DD  SYSOUT=(*)
//XPSL0001 DD  DISP=SHR,DSN=COMMON.DDIO                              NOTE 9
//XINCLUDE DD  DISP=SHR,DSN=SYS2.XPEDITER.V7R0M0.INCLUDE
//XPHELP   DD  DISP=SHR,DSN=SYS2.XPEDITER.V7R0M0.HELP
//XOPTIONS DD  DISP=SHR,DSN=SYS2.XPEDITER.V7R0M0.XOPTIONS
//XPSCRIPT DD  DISP=(NEW,PASS),UNIT=SYSDA,
//             SPACE=(TRK,(2,2))
//XPOUT    DD  DSN=&&XPLOG,
//             DISP=(MOD,PASS)
//XPSHOW   DD  DISP=(NEW,DELETE),UNIT=SYSDA,
//             DCB=(RECFM=FB,BLKSIZE=6160,LRECL=80),
//             SPACE=(TRK,(2,2))
//XPIN     DD  *
XPED TSO
TEST TRIMAIN
//PRNTLOG  EXEC PGM=IEBGENER,COND=EVEN
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DUMMY
//SYSUT1   DD   DSN=&&XPLOG,DISP=(OLD,PASS)
//SYSUT2   DD   SYSOUT=*
```

Figure 4-24. JCL to Run a Program With XPEDITER/TSO

Setting Up the Batch JCL

The numbers listed in Figure 4-24 on page 4-44 correspond to the following notes:

Notes:

1. TSO users should modify this to their TSO user ID.
2. Change to your site's standard values.
3. Add this step before the step that executes your program. The CREATE EXEC step creates the log dataset or output from this job.
4. Change to PGM=XPTSO to run interactive batch. Change to PGM=XPBATCH to run unattended batch. The region size may need to be increased. An XPEDITER/TSO session requires up to 500K more region than a session without XPEDITER/TSO. Keep the original parameter list, if any.
5. This DD statement creates a dynamic work area that XPEDITER/TSO requires to execute. Place it before the STEPLIB concatenation. This is a temporary PDS that will be deleted when the job ends. Change the BLKSIZE to be equal to the largest blocksize of any library in the STEPLIB concatenation for the test. This step is required.
6. In the STEPLIB ddname, concatenate the XPEDITER/TSO work dataset created by the XDYNAMIC ddname at the beginning of STEPLIB. The XPEDITER/TSO LOADLIB and the CSS LOADLIB must also be in the STEPLIB concatenation.
7. Change to your user LOADLIB.
8. Specify your library.
9. Add the following nine ddnames after the last standard ddname. Make sure the XPEDITER/TSO ddnames are allocated to the correct dataset names.

   XPSL000n DD
   DDIO dataset that contains the source listing members of the compiled modules.

   XINCLUDE DD
   Optional script file (PDS) to be used with the INCLUDE, INITSCR, and POSTSCR commands. This ddname can be omitted if your site does not support INCLUDE scripts or if INCLUDE, INITSCR, and POSTSCR are not used. If the Site-wide Script Library is desired, it should be first in the concatenation.

   XPHELP DD
   XPEDITER/TSO help dataset.

   XOPTIONS DD
   Enhanced FIND function parameters file.

   XPSCRIPT DD
   The generated script dataset. All XPEDITER input commands are recorded in this file. System responses are not recorded.

   XPOUT DD
   The output (log file) from this job. If running an interactive batch job, XPOUT must be directed to a dataset with DISP=MOD.

   If you direct the output of XPOUT ddname to a dataset rather than to SYSOUT, specify the disposition as MOD. The DCB attributes are:

   ```
   DCB=(RECFM=VBA,LRECL=137,BLKSIZE=6854)
   ```

   XPSHOW DD
   A temporary work file used in processing XPEDITER SHOW commands.

   XPIN DD
   Enter the commands to execute the test session. The first command must be XPED with the appropriate environment parameter. Then, the TEST command with the program name and any required parameters. When running an interac-

tive batch test, this is the only input required. The XPED and TEST commands are described in the *XPEDITER/TSO and XPEDITER/IMS Reference Manual*.

If running unattended batch, the debugging commands follow the TEST command. The debugging commands can be included as a command stream after the TEST command, or a dataset containing the commands can be specified on the TEST command. Refer to "Creating a Command Stream" for more information.

Usage Notes

1. If you are using a PROC, remember that JCL rules for PROCs prohibit the use of DD * inline data streams. For that reason, each reference to XPIN should be coded:

   ```
   //XPIN DD DUMMY
   ```

   or

   ```
   //XPIN DD DDNAME=SYSIN
   ```

   or omitted entirely.

   Then, in the actual JCL that invokes your new PROC, you must override the XPIN ddname with your inline XPED and TEST (or INTERCEPT for BTS) commands.

   When you execute the PROC, insert the //*stepname*.XPIN DD * statement in the override JCL.
2. If you are using a job that checks condition codes or abends for recovery or follow-up steps, be aware that XPEDITER/TSO usually produces a condition code of zero regardless of how the application program or XPEDITER/TSO actually ends.
3. No additional tasks are required to implement batch testing of programs compiled with CA-OPTIMIZER. However, it is important to note that XPEDITER/TSO dynamically turns off DTECT, PFLOW, and XCOUNT if specified. Therefore, a batch job run with both XPEDITER/TSO and CA-OPTIMIZER receives only XPEDITER/TSO debugging output.
4. **Even if your application programs would normally find any required Language dependent run-time subroutines (including LE - Language Environment), without being included in the JOBLIB/STEPLIB of the Batch JCL (usually from the LINKLIST or (E)LPA), the libraries must still be specified as part of the test session setup. This will ensure that XPEDITER's Task Library will be properly configured. For manually altered Batch JCL, you should include the run-time libraries in the STEPLIB DD statement concatenations of the JCL step(s) that were altered to be intercepted.**

Creating a Command Stream

Batch mode XPEDITER/TSO reads in a command stream from the XPIN DD, performs the debugging functions, and writes the results to the XPOUT DD. A command stream is a predefined set of XPEDITER/TSO commands stored in a PDS member or a sequential file, or included as instream data.

The first step in creating a command stream is to specify the environment in which the program is to be debugged. Enter the XPED command with the type of environment (for example, TSO corresponds to the interactive STANDARD environment).

The next step is to specify the name of the program to be debugged. Enter the TEST command along with the program name if the execution environment was specified as BATCH or IMS. Enter the INTERCEPT command along with the program name if the execution environment was specified as BTS.

Example 1:

```
XPED TSO
TEST TRIMAIN
```

Example 2:

```
XPED IMS
TEST TRIMAIN
```

Example 3:

```
XPED BTS
INTERCEPT TRIMAIN
```

The third step is to list all the debugging tasks to be performed during the batch debugging session. Under the BATCH or IMS environment, you can enter other XPEDITER/TSO commands following the TEST command; however, you can also alternatively execute a test script stored in a separate file.

All XPEDITER/TSO commands must be entered in uppercase when creating a command stream. At least one space is required to delimit the keyword from its value. Commands can be entered on one line or on consecutive lines.

Specifying Multiple Debugging Sessions Within the Same Job Stream

Although you cannot change environments during a debugging session, it is expected that you will debug various programs, thus TEST or INTERCEPT can be used as many times as necessary.

Multiple programs within the same environment can be debugged in one batch run. The TEST command is used to specify the program to be debugged in batch mode. Any number of debugging sessions can be run within a single job stream (except for IMS debugging sessions). All XPEDITER/TSO commands following a TEST command are executed in relation to the program specified by the TEST command—until another TEST or INTERCEPT command is encountered.

The INTERCEPT command is used to change module qualification. Any commands entered following the INTERCEPT command and before another TEST, INTERCEPT, or EXIT command are executed in relation to the program specified on the INTERCEPT command.

For BTS debugging, only the XPED and INTERCEPT commands are allowed within a command stream. All debugging session commands are included in a test script associated with the program specified on the INTERCEPT command.

Creating Batch JCL for XPEDITER/TSO Options

Database Support (Including IMS/DB and IMS/DB With DB2 Testing Under TSO)

To test your IMS/DB program with unattended or interactive batch, use the steps listed below:

1. Retrieve your IMS/DB program execution JCL.
2. Copy the member BATCHIMS from the SAMPLIB dataset and follow the instructions in it.
3. If you are using a PROC, change the XPIN ddname to:

   ```
   //XPIN DD DUMMY
   ```

4. When you are executing the PROC, insert the //*stepname*.XPIN DD * statement in the override JCL.

   For more information regarding the use of the //XPIN DD statement in a PROC, refer to "Usage Notes" on page 4-49.

5. Submit the job. If you are running unattended batch, the test results will be saved in the XPOUT file. If you are using interactive batch, you must connect to the job to complete testing.

BTS Support

To test your BTS program with unattended or interactive batch, use the steps listed below:

1. Retrieve your BTS program execution JCL.
2. Copy the member BATCHBTS from the SAMPLIB dataset and follow the instructions in it.
3. If you are using a PROC, change the XPIN ddname to:

   ```
   //XPIN DD DUMMY
   ```

4. When you are executing the PROC, insert the //*stepname*.XPIN DD * statement in the override JCL.

   For more information regarding the use of the //XPIN DD statement in a PROC, refer to "Usage Notes" on page 4-49.
5. Submit the job. If you are running unattended batch, the test results will be saved in the XPOUT file. If you are using interactive batch, you must connect to the job to complete testing.

DB2 Support

To test your DB2 program with unattended or interactive batch, use the steps listed below:

1. Retrieve your DB2 program execution JCL.
2. Copy the member BATCHDB2 from the SAMPLIB dataset and follow the instructions in it.
3. If you are using a PROC, change the XPIN ddname to:

   ```
   //XPIN DD DUMMY
   ```

4. When you are executing the PROC, insert the //*stepname*.XPIN DD * statement in the override JCL.

   For more information regarding the use of the //XPIN DD statement in a PROC, refer to "Usage Notes" on page 4-49.
5. Submit the job. If you are running unattended batch, the test results will be saved in the XPOUT file. If you are using interactive batch, you must connect to the job to complete testing.

Test Session Management Using Scripts

A script is a predefined stream of XPEDITER/TSO commands that can be used to:

1. Initialize a debugging session (*initial script*).
2. Execute a set of commands during the debugging session (*test script*).
3. Execute a set of commands at the end of a session (*post script*).
4. Execute a set of commands when an abend occurs in an unattended batch session (*abend script*).

Scripts enable you to:

- Eliminate redundant keystrokes.
- Play back the commands established during a previous session.
- Run regression testing.

You create and maintain the scripts and the script libraries you use. A script library is a partitioned dataset (PDS) that can be FB 80 (fixed block with a record length of 80) or a VB 255 (variable blocked with a maximum record length of 255). If you use FB 80, only the data in columns 1 through 72 is recognized.

If you intend to use a script during the test session, the script library must be preallocated before the session begins. Use the SETUP command and select Option 3 (INCLUDES) on the Setup Menu to allocate the library, unless the script member is contained in a Site-wide library specified at installation time. Refer to “Saving and Using Generated Scripts” on page 4-45 for steps on using a generated script.

Initial Scripts

An initial script is executed at the *beginning* of the debugging session or at the before breakpoint of each module. It can be used to:

- Set initial PF key values.
- Specify XPEDITER/TSO debugging session processing options with the SET command.
- Set initial breakpoints in modules.
- Initialize data items within the program (using MOVE command).

In interactive mode, the initial script member name is specified in the Initial Script field on the selected environment test screen.

In batch connect interactive or unattended mode, the initial script member name is specified in the INITSCR field on the Select Job Step screen, or you can edit your execution JCL and use the INITSCR parameter on the TEST or INTERCEPT command.

The ability to process SET commands before the program is loaded is the most useful feature of the initial script. The following SET commands either configure XPEDITER/TSO or control the manner in which XPEDITER/TSO loads programs:

```
SET CBLTRAP              SET STATIC
SET DYNAMIC              SET TRANSFER
SET EXCLUDE
```

The following commands determine the size of the log and the way information is represented.

```
SET HEXMODE
SET LOGSIZE
SET NONDISP
```

Refer to the *XPEDITER/TSO and XPEDITER/IMS Reference Manual* for further information regarding the use of the SET command parameters.

Test Scripts

A test script is used to execute XPEDITER/TSO debugging commands *during* a debugging session. The commands in the test script are executed in the order they are read, as if they had been entered serially from the terminal.

In interactive mode or batch connect interactive, use the INCLUDE command with the test script member name at any time within the session to execute a test script command stream.

In unattended batch, the test script member name is specified by using the INCLUDE command in the JCL job stream.

In interactive mode, XPEDITER/TSO automatically generates a test script of all the commands entered during the debugging session. The generated file can be edited and copied into a script library for later use when you want to duplicate the debugging session. The generated test script is accessed by typing the SCRIPT command on the command line of the Test screen after a debugging session. Refer to “Saving and Using Generated Scripts” on page 4-45 for additional information.

Post Scripts

A post script is comprised of a command or set of commands that are executed when the end of the program is encountered. The commands are effectively executed *after* the debugging session is ended, but before the XPEDITER/TSO environment is exited.

A post script has many purposes. For instance, it lets you display (PEEK) the value of variables at the close of the debugging session and show the COUNT tallied on program statements.

In interactive mode, a post script member name is specified in the Post Script field on the selected environment test screen.

In batch connect interactive or unattended mode, a post script member name is specified by editing your execution JCL and using the POSTSCR parameter on the TEST or INTERCEPT command.

Notes:

1. It is suggested that you use PEEK instead of KEEP in a post script.
2. When debugging VS COBOL II programs compiled with DATA(31) and RENT, and linked with AMODE(31) and RMODE(ANY), post scripts cannot be used to execute the PEEK, KEEP, and MOVE commands.

Abend Scripts

An abend script is an XPEDITER/TSO command or set of commands that are executed when an abend occurs. You can use an abend script only when debugging in unattended mode.

Use the SET ABENDSCR command in the JCL job stream to specify the abend script to be executed when an abend occurs. The commands included in that script are executed whenever an abend occurs until another SET ABENDSCR command is executed. Any number of SET ABENDSCR commands can be included in a job stream.

Note: It is suggested that you use PEEK instead of KEEP in an abend script.

Creating and Editing Scripts

In a script, the following must be observed:

- Commands cannot exceed 100 characters.
- Commands must be entered in uppercase. Lowercase characters are not valid.
- Only one XPEDITER/TSO command can be entered on a single line.
- A command can be continued beyond a single line without a continuation character.
- A quoted string must be contained on one line.
- Test scripts can be nested without limit by inserting additional INCLUDE commands within the test script.
- Comment lines can be included by entering an asterisk (*) in column 1.

- When used within inserted code, the INCLUDE command is executed when the inserted code is executed.

Scripts are extremely useful under certain debugging circumstances. For example, suppose you end a debugging session knowing that at some later time you intend to retest the program along the same lines. A script is an efficient way to quickly reproduce that session without having to reenter the commands individually.

Saving and Using Generated Scripts

In interactive mode, XPEDITER/TSO automatically generates a test script of all the commands entered during the debugging session. The following steps will show you how to save and use the script for later use when you want to duplicate the debugging session.

Option 1 - Batch Connect Under XPEDITER/TSO

The following steps apply to XPEDITER/TSO only:

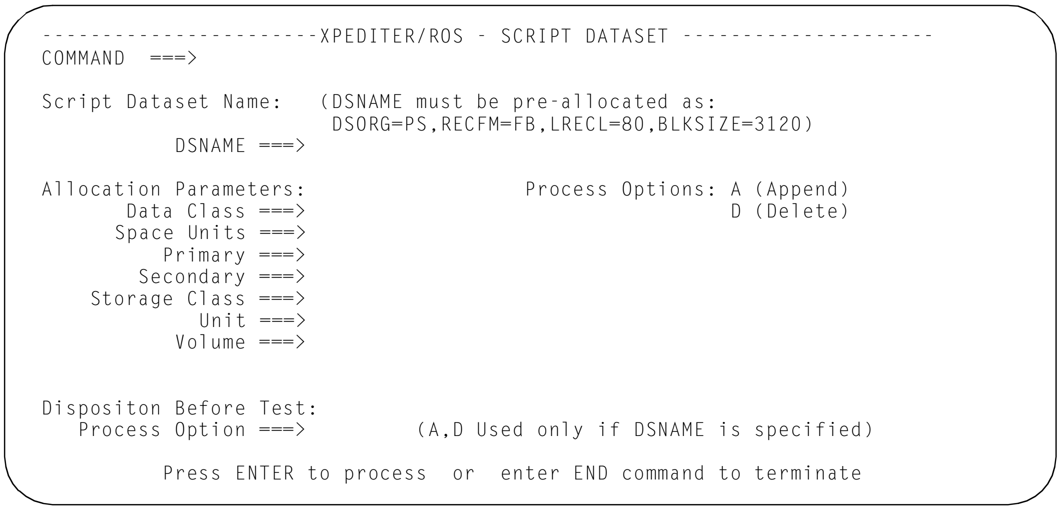
1. When the debugging session is completed, the message "Log and Script Created" appears in the upper right corner of the Test screen. Enter **SCRIPT** on the command line of the Test screen.
2. Edit the displayed script according to the guidelines in “Creating and Editing Scripts” on page 4-44.
3. Press **PF3** to save the edited script.
4. On the Data Set Disposition Screen, enter **C** in the Process Option field to copy the edited script to a partitioned dataset (PDS).
5. If you have already allocated a PDS for this purpose, enter the PDS name in the DSNAME field of this screen and a name for the edited script in the Member Name field. Press **Enter** and skip to Step 7. Otherwise, go to the next step.
6. If you have not allocated a PDS, you can do so at this time by entering a library name in the DSNAME field of the screen and a name for the edited script in the Member Name field. Press **Enter** and the New Dataset Allocation screen is displayed with the dataset name you selected prefilled in the Dataset Name field. Enter the parameters for the dataset according to your site standards and press **Enter** to process.
7. You are returned to the Test screen and a message specifying the number of lines that were copied is displayed in the upper right corner of the screen. If your PDS is not specified in setup on the Test Script Libraries screen, enter **SETUP** from the Test screen and enter **3** on the Setup Menu screen. Then enter the PDS name on the Test Script Libraries screen. Press **Enter** to process.
8. Press **PF3** to return to the Test screen.
9. When you want to use the script for a subsequent test session, specify the PDS member name of the script in the Test screen’s Initial Script field or use the INCLUDE command followed by the PDS member name to execute the script after the source is displayed.

   Note: The LOG and SCRIPT commands are still displayed on the Test screen. This means that the log and script files from the previous session are still active. If you intend to start another debugging session and want to use fresh log and script files, press **PF3** to return to the XPEDITER/TSO Primary Menu and then enter option **2** to return to the Test screen. This deletes the previous log and script files, and you are ready to begin a new session.

Option 2 - Batch Connect Under ROSCOE

The following steps apply when using the XPEDITER/ROSCOE Batch Connect Environment:

1. Create a dataset that will be used in the test session while the batch connect job is running. This dataset should be allocated as follows: DSORG=PS, RECFM=FB, LRECL=80, and BLKSIZE=3120
2. Go to the SETUP menu on any of the ROSCOE Front End screens and go to the OPTION field and select **6** from the menu choices.
3. Press **Enter** and you will view the ROSCOE SCRIPT DATASET screen as displayed in Figure 4-25.



```
----------------------XPEDITER/ROS - SCRIPT DATASET --------------------
COMMAND  ===>

Script Dataset Name:   (DSNAME must be pre-allocated as:
                        DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=3120)
           DSNAME ===>

Allocation Parameters:                  Process Options: A (Append)
       Data Class ===>                                   D (Delete)
      Space Units ===>
          Primary ===>
        Secondary ===>
    Storage Class ===>
             Unit ===>
           Volume ===>


Dispositon Before Test:
   Process Option ===>         (A,D Used only if DSNAME is specified)

          Press ENTER to process  or  enter END command to terminate
```

Figure 4-25. XPEDITER/ROSCOE SCRIPT DATASET

4. Specify the dataset name allocated in Step 1 and set the process option based on whether you wish to append or delete the old content.
5. Run your test and upon completion, copy the data from the dataset on the XPSCRIPT to a member in your INCLUDE dataset.
6. From this point forward, you need to include the script on the Select Job Step screen as your initial script or you must issue the INCLUDE script name while in the source.

Script Examples

Example 1:

Suppose you are debugging TRIRPT as a stand-alone program. To do so, you need to MOVE values that have been passed from TRIMAIN to variables in the Linkage Section. The Linkage Section of TRIRPT is shown in Figure 4-26.

```
000028    LINKAGE SECTION.
000029    01  TABLE-OF-NAMES-N-CNTRS.
000030        05  N-N-C                OCCURS 4 TIMES
000031                                 INDEXED BY TX.
000032            10 T-NAME            PIC X(21).
000033            10 T-CNTR            PIC 9(04).
```

Figure 4-26. Linkage Section of the TRIRPT Program

In this case, use a script to move valid data to the variables in the Linkage Section before beginning the debugging session. For example, the script shown in Figure 4-27 on page 4-47 moves valid data to the table variables and counter. This script can be executed as a test script with the INCLUDE command within a debugging session, or the MOVE commands can be made part of an initial script command stream executed at the outset of the debugging session.

```
    MOVE 'EQUILATERAL TRIANGLES' TO T-NAME(1)
    MOVE 'ISOSCELES TRIANGLES' TO T-NAME(2)
    MOVE 'SCALENE TRIANGLES' TO T-NAME(3)
    MOVE 'INVALID TRIANGLES' TO T-NAME(4)
    MOVE 2 TO T-CNTR(1)
    MOVE 3 TO T-CNTR(2)
    MOVE 6 TO T-CNTR(3)
    MOVE 4 TO T-CNTR(4)
```

Figure 4-27. Sample Script Containing MOVES to Pass Values From TRIMAIN to Variables in TRIRPT

Example 2:

Suppose you want to set a number of breakpoints that are sufficiently spaced apart and require you to scroll up after setting each breakpoint. A script comprised of these breakpoint commands could initialize the session, setting the breakpoints. This would eliminate the need to scroll in order to enter the commands. Such a script is illustrated in Figure 4-28.

```
    BEFORE 34  41  TRIMAIN:
    AFTER  40  94  132  TRITST:  TRIRPT:WRITE-DTLS
    COUNT  43  55  93  101
    SKIP   98 THRU 134   WRITE-REPORT
```

Figure 4-28. Sample Script

Using XPEDITER/TSO Commands in a Script (Interactive)

Refer to the INCLUDE command in the *XPEDITER/TSO and XPEDITER/IMS Reference Manual* for a list of the commands that can and cannot be included in a script for interactive debugging of COBOL programs.

Using XPEDITER/TSO Commands in a Script (Unattended)

Since all of the commands entered in a script for unattended debugging are executed before the first statement in the program is encountered, careful planning of the debugging session is critical. Here are some considerations:

- Store all your command streams as members of your INCLUDE library. Then you are able to keep track of all data streams you have used. It also saves time by not having to reenter all the commands for each session.
- AFTER, BEFORE, and TRACE commands give approximately the same information. All three update the Keep window.

  When an AFTER command is executed and changes are logged, the log entry for the command identifies the statements before and after the place where execution is paused. Therefore, use of the AFTER command is a good means of tracking the flow of execution. It is particularly useful following a conditional statement when you are uncertain of the execution path.
- To track a value change in a variable, using the WHEN command is better than using the TRACE command, which produces a large amount of output in the log file.
- PEEK is an immediate command, which means the display to the log occurs as soon as the PEEK command is encountered. PEEK is most useful if you want to see a field at the very beginning or at the very end of a program, and you only want to see the value once. It can be useful within inserted code. You could insert a PEEK after the statement you want executed. The PEEK command would be executed each time the statement was executed.
- In general, the KEEP command is used more often than the PEEK command because a keep causes the value of a data item to be displayed repeatedly. KEEP displays a value only at the location of a breakpoint and only when a value has changed. Therefore the best way to watch data change within a program is to set breakpoints (AFTER,

BEFORE, or TRACE) on key statements and keep all the data items you want to monitor.

CAUTION
In post and abend scripts, use the PEEK command instead of the KEEP command.

-
- The COUNT command can be used to monitor statement execution in a batch program that processes large amounts of data. To see the results of the COUNT command, issue the SHOW COUNTS command in a post script at the end of the debugging session.
- One of the problems with unattended debugging is that normally all commands need to be executed at the beginning of the debugging session. This can create more output than the user needs. An INSERT command allows XPEDITER/TSO commands to have delayed execution. The XPEDITER/TSO commands within an INSERT are not executed until the inserted code is executed; that is, when the previous statement is executed. INSERT can be a very powerful command in an unattended batch debugging session.

Refer to the INCLUDE command in the *XPEDITER/TSO and XPEDITER/IMS Reference Manual* for a list of the commands that can and cannot be used in a script for an unattended debugging session for COBOL programs.

Accessing Other Systems From XPEDITER/TSO

Accessing File-AID for DB2

This section describes how to access and exit File-AID for DB2 from XPEDITER/TSO. File-AID for DB2 is Compuware's interactive DB2 database management tool that lets you create, view, change, and customize the DB2 table data associated with your program without coding SQL. File-AID for DB2 also provides facilities for interactive SQL development and analysis and a host of utilities. Refer to the *File-AID for DB2 Reference Manual* for information about using File-AID for DB2.

XPEDITER/TSO's DB2 capabilities are greatly enhanced when you have XPEDITER for DB2 Extension and File-AID for DB2 installed at your site. This versatile union lets you debug, prototype SQL calls, and analyze COBOL DB2 programs in XPEDITER/TSO. At the same time, you are also permitted to dynamically access File-AID for DB2, Compuware's interactive DB2 database management tool that lets you create, view, change, and customize the DB2 table data associated with the program without coding SQL. File-AID for DB2 also provides facilities for interactive SQL development and analysis and a host of utilities. Refer to "Using XPEDITER for DB2 Extension" on page 5-40 for information about how to browse and edit DB2 table data while testing your program, analyze SQL statement execution with the FADB2 EXPLAIN command, and prototype SQL logic by inserting SQL statements.

Notes:

1. XPEDITER/TSO supports debugging of any DB2 program within all execution environments selectable under the Environments Menu, with the following exceptions:
   - The first exception to this XPEDITER/TSO support rule is the XPEDITER/TSO dialog environment.
   - A second exception to normal debugging is that XPEDITER for DB2 Extension cannot be accessed when using the batch connect facility.
2. You must bind your program application plan with File-AID for DB2 if any of the following apply:

  - The program executes statically compiled SQL statements. If the program executes only dynamically inserted SQL statements, you can either bind your application plan with File-AID for DB2 or use the File-AID for DB2 default plan in place of your application plan.
  - The program executes in the IMS environment.

Refer to Appendix F, “Binding the Application Plan” for information about binding.

The following accesses File-AID for DB2 from XPEDITER/TSO:

- Select the FADB2 menu item on the XPEDITER/TSO Primary Menu.
- Enter the FADB2 command on any XPEDITER/TSO Source display screen.

The File-AID for DB2 Primary Option Menu is displayed. The menu lists all File-AID for DB2 facilities.

Notes:

1. The SQL Development and Analysis facility is available from XPEDITER/TSO only when File-AID for DB2 is accessed by entering **F** (FADB2) on the XPEDITER/TSO Primary Menu.
2. The plan name and DB2 subsystem you specified to start your XPEDITER/TSO test session are also in effect when you invoke File-AID for DB2 from an XPEDITER/TSO Source display screen. However, a subsystem switch is not permitted under File-AID for DB2 when it is accessed from XPEDITER/TSO.

You can bypass the File-AID for DB2 Primary Option Menu by entering the FADB2 command with the keyword or 1-character number or letter that directly accesses a specific File-AID for DB2 function. For example, to directly access the File-AID for DB2 Browse function from XPEDITER/TSO, enter:

```
FADB2 BROWSE
```

The keywords and 1-character numbers or letters that access File-AID for DB2 functions are listed on the File-AID for DB2 Primary Menu.

Exiting File-AID for DB2

When you finish using File-AID for DB2 functions, you can either:

1. Enter `=X` to return to the XPEDITER/TSO Source display screen displaying your source program.
2. Enter **END**, which will take you to the File-AID for DB2 Primary Option Menu from which you can select another File-AID for DB2 function.

Connecting to CICS

Option **4** (CICS) on the Primary Menu displays the TSO to CICS Connect Support screen shown in Figure 4-29 on page 4-50. This screen lets you select a CICS system and subsequently connect to a CICS session. It also lets you specify a PF key that enables you to easily return to TSO.

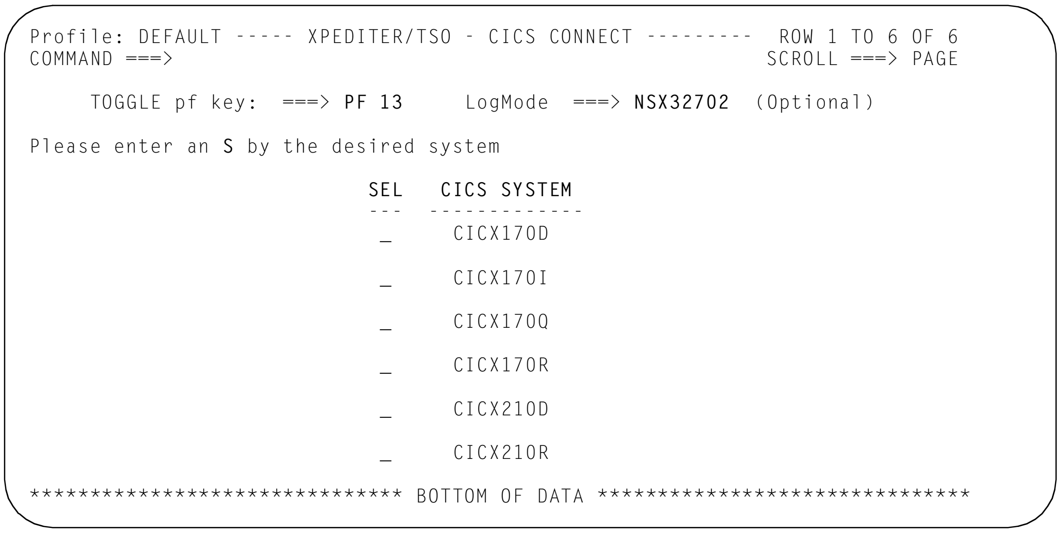

```
Profile: DEFAULT ----- XPEDITER/TSO - CICS CONNECT ---------  ROW 1 TO 6 OF 6
COMMAND ===>                                                 SCROLL ===> PAGE

     TOGGLE pf key:  ===> PF 13     LogMode  ===> NSX32702  (Optional)

Please enter an S by the desired system

                            SEL   CICS SYSTEM
                            ---  -------------
                             _     CICX170D

                             _     CICX170I

                             _     CICX170Q

                             _     CICX170R

                             _     CICX210D

                             _     CICX210R

****************************** BOTTOM OF DATA ******************************
```

Figure 4-29. TSO to CICS Connect Screen

Specify a PF key that you can use to toggle from CICS to TSO. Then, enter **S** in the SEL area of the CICS system you want to access. Press **Enter** and the CICS-VS screen of the selected region is displayed.

Optionally, a logmode can be entered in the LogMode field and XPEDITER/TSO will attempt to connect to CICS using the indicated mode. To use the default VTAM defined logmode, clear the LogMode field.

To return to TSO, press the selected PF key from any screen in your CICS session. When a currently connected CICS session is ended, the message `SESSION TERMINATED` appears in the upper right corner of the CICS Connect screen.

Chapter 5. Debugging Interactively

This chapter describes how to use XPEDITER/TSO to interactively debug your COBOL programs. It discusses:

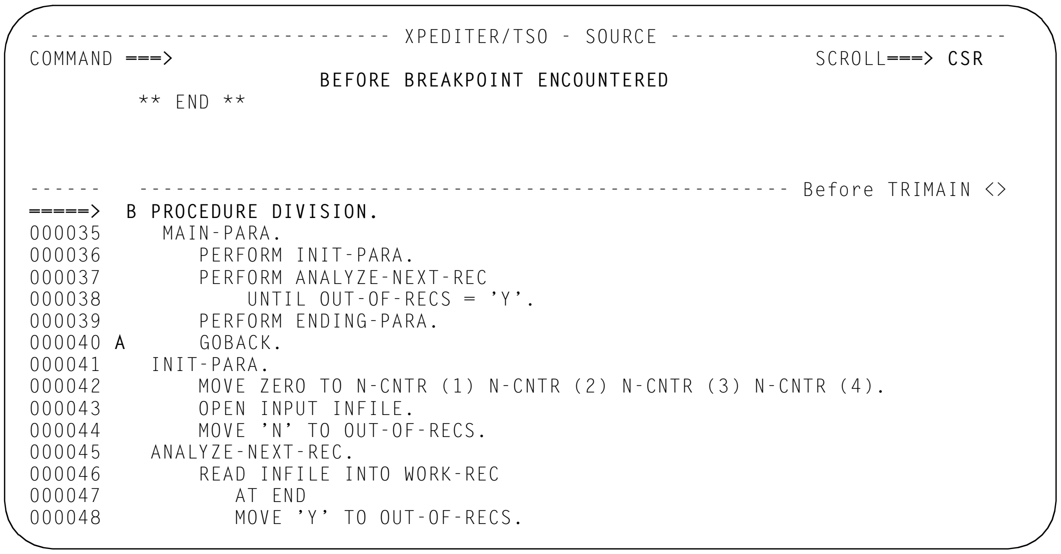
- The source display.
- Using XPEDITER/TSO commands.
- Qualification Rules for XPEDITER/TSO commands.
- Controlling execution and setting breakpoints with the BEFORE, AFTER, GO, INTERCEPT, RETURN, RETEST, PAUSE, TRACE, AT, COUNT MAX, SKIP, SOURCE, and WHEN commands.
- Inspecting and manipulating data (variables, registers, and memory) with the Automatic Keep function and the KEEP, PEEK, MOVE, MEMORY, and GPREGS commands.
- Analyzing program logic (data flow, execution path, and execution coverage) using the FIND, EXCLUDE, MONITOR, REVERSE, RESUME, TRACE, SHOW PREV, LOCATE, and COUNT commands.
- Modifying program logic with the SKIP, INSERT, GOTO, and MOVE commands.
- Displaying file allocations with the SHOW command.
- Translating and expanding EXEC statements using the GEN command.
- Debugging and testing programs that do not have the source (no source listing) available with the AT command.
- Displaying subsystem-related debugging information for VSAM, IDMS, IMS, and DB2 and environment specific run-time characteristics during a test session. Available only if you have Abend-AID release 7.0.2 or above.
- Using test scripts (a predefined stream of XPEDITER/TSO commands) to automate the test session (INCLUDE command).

In addition, XPEDITER/TSO supports debugging of any DB2 program within any execution environment selectable under the Environments Menu.

If you have XPEDITER for DB2 Extension and File-AID for DB2 installed at your site, you can also interactively insert SQL statements in your program, retrieve Explain information about SQL statements, and browse and edit DB2 table data. A series of FADB2 commands are available for these functions. Refer to “Using XPEDITER for DB2 Extension” on page 5-40 for more information.

The Source Display

When you start a debugging session, XPEDITER/TSO displays the program source listing as shown in Figure 5-1 on page 5-2.

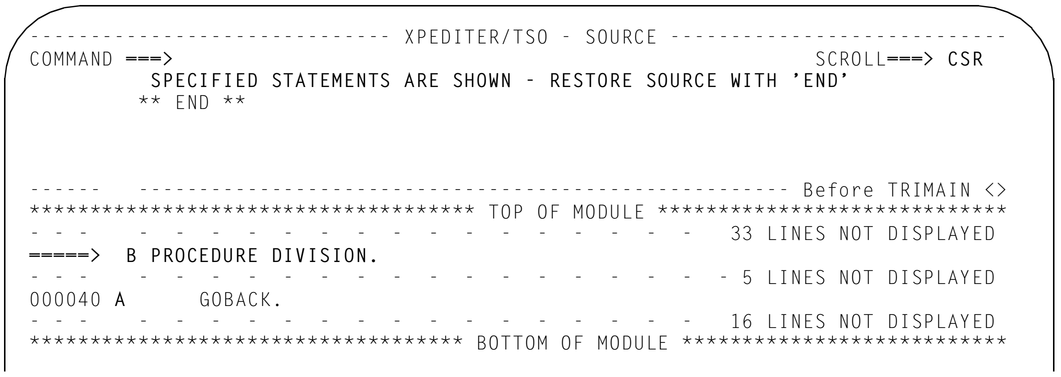


```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                     SCROLL===> CSR
                        BEFORE BREAKPOINT ENCOUNTERED
         ** END **



------   ---------------------------------------------------- Before TRIMAIN <>
=====>  B PROCEDURE DIVISION.
000035     MAIN-PARA.
000036        PERFORM INIT-PARA.
000037        PERFORM ANALYZE-NEXT-REC
000038            UNTIL OUT-OF-RECS = 'Y'.
000039        PERFORM ENDING-PARA.
000040 A      GOBACK.
000041    INIT-PARA.
000042        MOVE ZERO TO N-CNTR (1) N-CNTR (2) N-CNTR (3) N-CNTR (4).
000043        OPEN INPUT INFILE.
000044        MOVE 'N' TO OUT-OF-RECS.
000045    ANALYZE-NEXT-REC.
000046        READ INFILE INTO WORK-REC
000047           AT END
000048           MOVE 'Y' TO OUT-OF-RECS.
```

Figure 5-1. XPEDITER/TSO Source Display Screen

XPEDITER/TSO automatically sets a before breakpoint on the PROCEDURE DIVISION statement, and an after breakpoint on the GOBACK or STOP RUN statement.

Figure 5-2 shows an example of entering the **SHOW BREAKS** command for the program above.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                     SCROLL===> CSR
          SPECIFIED STATEMENTS ARE SHOWN - RESTORE SOURCE WITH 'END'
         ** END **



------   ---------------------------------------------------- Before TRIMAIN <>
************************************ TOP OF MODULE ****************************
- - -    - - - - - - - - - - - - - - - - - - - - - - -   33 LINES NOT DISPLAYED
=====>  B PROCEDURE DIVISION.
- - -    - - - - - - - - - - - - - - - - - - - - - - - - 5 LINES NOT DISPLAYED
000040 A      GOBACK.
- - -    - - - - - - - - - - - - - - - - - - - - - - -   16 LINES NOT DISPLAYED
********************************** BOTTOM OF MODULE ***************************
```

Figure 5-2. Result of Using the SHOW BREAKS Command

The execution status field on the ninth line indicates the current execution position and the <> at the end of line indicates that the screen can be scrolled. An arrow points to the statement where execution is paused. The breakpoint indicator areas in column 9 and columns 74 through 80 show the breakpoints that have been set.

Using XPEDITER/TSO Commands

XPEDITER/TSO commands are used to interactively test and debug your programs and to perform other functions associated with your test session. Some XPEDITER/TSO commands are primary commands that must be entered on the command line. In many cases, a command will also have a corresponding line command that can be entered in the designated line command area on a screen.

Please refer to the *XPEDITER/TSO and XPEDITER/IMS Reference Manual* for detailed information about each command.

Controlling Program Execution

This section describes the XPEDITER/TSO commands that enable you to control program execution. Some of these commands, called *breakpoint commands*, are used to start and stop program execution at any time or when a specified condition occurs. Breakpoint commands let you designate the location of the pause by statement number, paragraph name, or module name, or by the occurrence of a particular event. The other commands that control program execution are used to specify the program to be displayed as the active program.

When program execution is paused, you can enter other XPEDITER/TSO debugging commands to examine program data, analyze and follow program logic, and many other debugging functions. Execution resumes when you enter **GO** or press **PF12**.

Program execution is automatically stopped when an abend occurs. XPEDITER/TSO intercepts program abends and automatically pauses at the failing statement. Some abends can be corrected dynamically, and execution can be resumed without terminating the session.

Note: If you wish to debug your EXEC DLI and EXEC SQL statements at the expanded level, place the letter **G** (GEN) on the line command area next to the statement and press **Enter** to display the expanded instructions. You can then set breakpoints at each generated statement.

Commands That Control Program Execution

The following are XPEDITER/TSO commands that control program execution.

Note: Breakpoint commands can be removed with the DELETE command.

AFTER	Stops program execution *after* the specified line of code is executed. An after breakpoint is automatically set on each GOBACK or STOP RUN statement in the driver program before the program begins executing.
AT	Sets a breakpoint at a sourceless main program or subprogram. Sourceless debugging and using the AT command are discussed in “Debugging a Sourceless Program” on page 5-46.
BEFORE	Stops program execution *before* the specified line of code is executed. A before breakpoint is automatically set on the PROCEDURE DIVISION statement in the driver program before the program starts executing. You must set a before breakpoint at the beginning of a called module if you want to stop execution before the called module executes.
COUNT	Counts the number of times an instruction or paragraph is executed and suspends program execution upon reaching a limit.
GO	Begins program execution and resumes execution after a pause.
INTERCEPT	Loads the specified module and sets a before breakpoint at the PROCEDURE DIVISION before the program begins executing and an after breakpoint on the GOBACK, STOP RUN, or EXIT PROGRAM.
PAUSE	Pauses execution within a block of code that has been inserted.
RETEST	Reloads the program you are debugging and restarts the test session.
SKIP	Bypasses execution of source code, which can include bypassing a module. The SKIP command adds flexibility to your testing by letting you skip instructions that you do not want executed.
SOURCE	Loads the specified module and makes it the active program. Any subsequent XPEDITER/TSO commands that are entered are applied to the specified module.

TRACE	Traces the statements as it executes and suspends execution upon reaching a set limit, when attention has been requested, or when terminal IO has been done.
WHEN	Suspends execution when a data value changes or when a specified condition occurs.

Entering Program Control Commands

A simple way to enter a breakpoint command is to scroll to the source line where you want execution to pause, type the command in the line command area, and press **Enter**. You will see an indication of the breakpoint in column 9 on the Source screen.

To set a breakpoint at a specific location or occurrence of a specified event, enter the breakpoint command from the primary command line with a location operand or condition for the stop. Refer to the *XPEDITER/TSO and XPEDITER/IMS Reference Manual* for the valid location operands and parameters for a command.

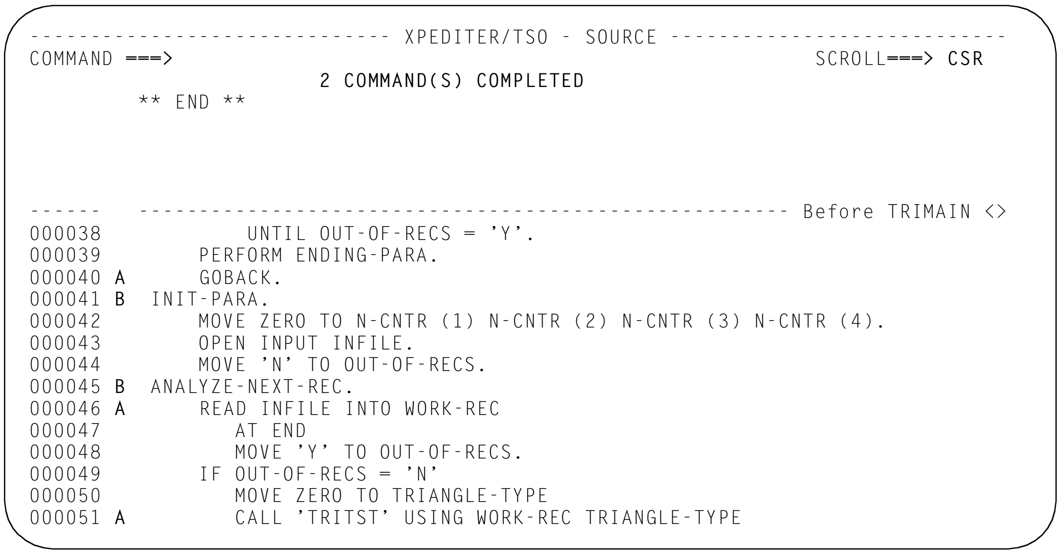
If the primary command line is too small to list breakpoints or to stack commands, enter the **SET CMDSIZE** command to expand the command line up to three lines.

Setting Before and After Breakpoints

The following commands set before breakpoints on every paragraph and after breakpoints on statements 46 and 51:

```
BEFORE ALL PARA;AFTER 46 51
```

When you press **Enter**, the screen looks like Figure 5-3.



```
----------------------------- XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                     SCROLL===> CSR
                       2 COMMAND(S) COMPLETED
         ** END **




------   ------------------------------------------------------- Before TRIMAIN <>
000038               UNTIL OUT-OF-RECS = 'Y'.
000039           PERFORM ENDING-PARA.
000040 A         GOBACK.
000041 B     INIT-PARA.
000042           MOVE ZERO TO N-CNTR (1) N-CNTR (2) N-CNTR (3) N-CNTR (4).
000043           OPEN INPUT INFILE.
000044           MOVE 'N' TO OUT-OF-RECS.
000045 B     ANALYZE-NEXT-REC.
000046 A         READ INFILE INTO WORK-REC
000047              AT END
000048              MOVE 'Y' TO OUT-OF-RECS.
000049           IF OUT-OF-RECS = 'N'
000050              MOVE ZERO TO TRIANGLE-TYPE
000051 A            CALL 'TRITST' USING WORK-REC TRIANGLE-TYPE
```

Figure 5-3. Result of Setting BEFORE ALL PARA and AFTER 46 51 Breakpoints

The after breakpoint on statement 46 causes execution to pause after the READ statement is executed and before the next statement is executed—statement 48 or 49, depending on the AT END condition.The after breakpoint on statement 51 causes execution to pause after the called module TRITST returns to the calling module TRIMAIN.

To set a breakpoint in a module that is not currently displayed, you must qualify the breakpoint by entering the module name terminated with a colon (:) before the breakpoint name.

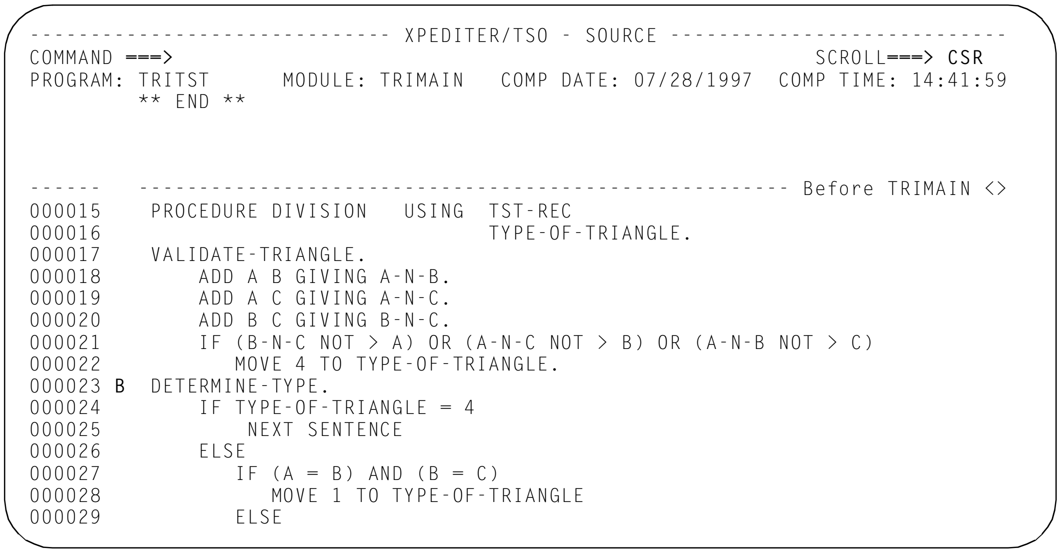
Alternatively, you can first display the source of the program (SOURCE command) that will be called later to establish module qualification, and then set breakpoints at the desired locations. Refer to the *XPEDITER/TSO and XPEDITER/IMS Reference Manual* for information about qualification rules for XPEDITER/TSO commands.

If the program is a member of a statically-linked module that is not yet loaded in memory, bring the module into storage by entering the **LOAD** *module-name* command first, then the **SOURCE** *program-name* command.

The following demonstrates using a qualified before breakpoint on paragraph DETERMINE-TYPE in module TRITST, and using the SOURCE command to display the program TRITST to verify that the before breakpoint is set correctly.

```
BEFORE TRITST:DETERMINE-TYPE;SOURCE TRITST
```

When you press **Enter**, the screen looks like Figure 5-4.



```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                       SCROLL===> CSR
PROGRAM: TRITST      MODULE: TRIMAIN   COMP DATE: 07/28/1997  COMP TIME: 14:41:59
         ** END **



------   ------------------------------------------------------ Before TRIMAIN <>
000015    PROCEDURE DIVISION   USING  TST-REC
000016                                TYPE-OF-TRIANGLE.
000017    VALIDATE-TRIANGLE.
000018        ADD A B GIVING A-N-B.
000019        ADD A C GIVING A-N-C.
000020        ADD B C GIVING B-N-C.
000021        IF (B-N-C NOT > A) OR (A-N-C NOT > B) OR (A-N-B NOT > C)
000022           MOVE 4 TO TYPE-OF-TRIANGLE.
000023 B  DETERMINE-TYPE.
000024        IF TYPE-OF-TRIANGLE = 4
000025            NEXT SENTENCE
000026        ELSE
000027           IF (A = B) AND (B = C)
000028              MOVE 1 TO TYPE-OF-TRIANGLE
000029           ELSE
```

Figure 5-4. Result of Entering Qualified Breakpoint

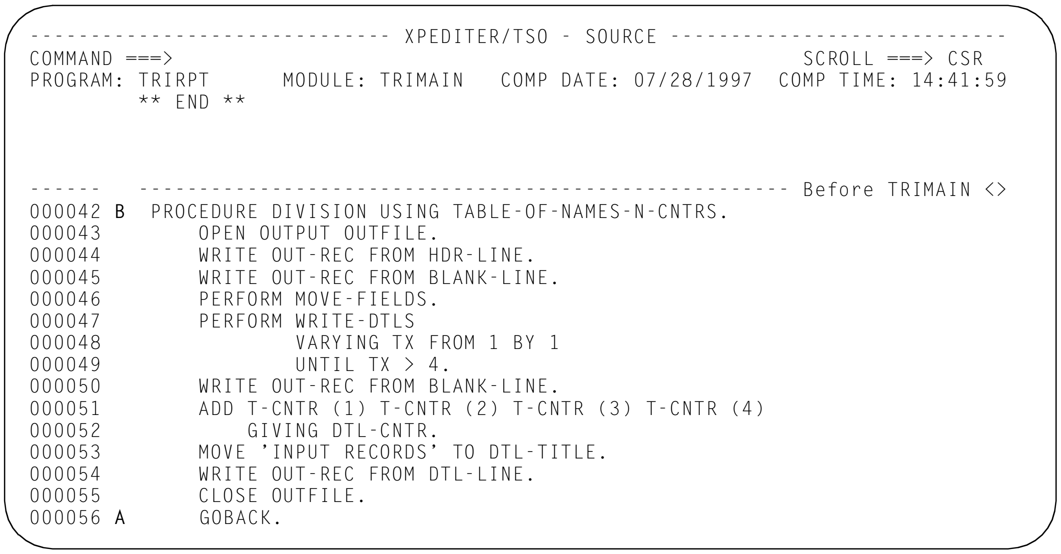
Enter **SOURCE** without a keyword or press **PF6** (LOCATE *) to display the active program (TRIMAIN) where execution is currently paused.

Using the INTERCEPT Command

The INTERCEPT command loads the specified module and sets before and after module breakpoints. The command performs the function of the AFTER module breakpoint, BEFORE module breakpoint, and the SOURCE commands combined. The following demonstrates using the INTERCEPT command to access module TRIRPT.

```
INTERCEPT TRIRPT
```

When you press **Enter**, the screen looks like Figure 5-5 on page 5-6.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                     SCROLL ===> CSR
PROGRAM: TRIRPT      MODULE: TRIMAIN   COMP DATE: 07/28/1997  COMP TIME: 14:41:59
         ** END **



------   ---------------------------------------------------- Before TRIMAIN <>
000042 B  PROCEDURE DIVISION USING TABLE-OF-NAMES-N-CNTRS.
000043        OPEN OUTPUT OUTFILE.
000044        WRITE OUT-REC FROM HDR-LINE.
000045        WRITE OUT-REC FROM BLANK-LINE.
000046        PERFORM MOVE-FIELDS.
000047        PERFORM WRITE-DTLS
000048                VARYING TX FROM 1 BY 1
000049                UNTIL TX > 4.
000050        WRITE OUT-REC FROM BLANK-LINE.
000051        ADD T-CNTR (1) T-CNTR (2) T-CNTR (3) T-CNTR (4)
000052            GIVING DTL-CNTR.
000053        MOVE 'INPUT RECORDS' TO DTL-TITLE.
000054        WRITE OUT-REC FROM DTL-LINE.
000055        CLOSE OUTFILE.
000056 A      GOBACK.
```

Figure 5-5. Result of the INTERCEPT Command

You can reset all the breakpoints you have entered since the beginning of your debugging session by issuing the RETEST command to obtain a “fresh” copy of the program.Or, you can resume execution of your program by issuing the GO command.

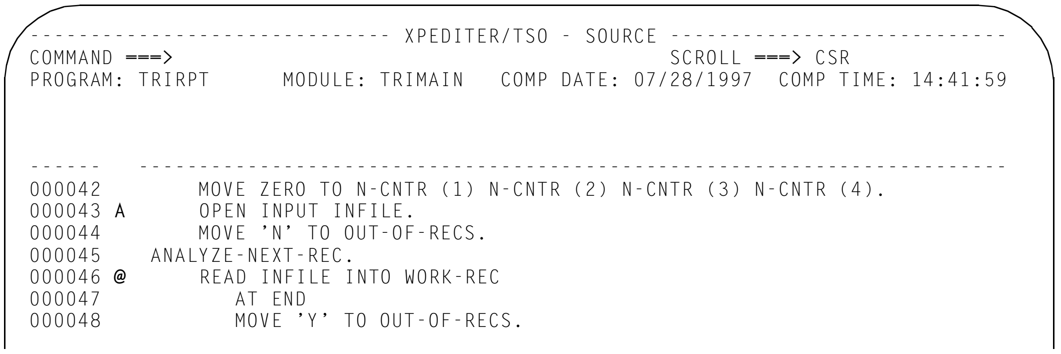
Resuming Execution With the GO Command

The GO command (PF12/PF24) is used to begin or resume execution of your program. Your program will execute until a breakpoint is encountered, an abend is intercepted by XPEDITER/TSO, or the end of the program is reached.

For example, in the program TRIMAIN, program execution paused when the before breakpoint was encountered on the PROCEDURE DIVISION.Suppose you entered the following command stream to set explicit after and before breakpoints:

```
AFTER 43 46;BEFORE 46
```

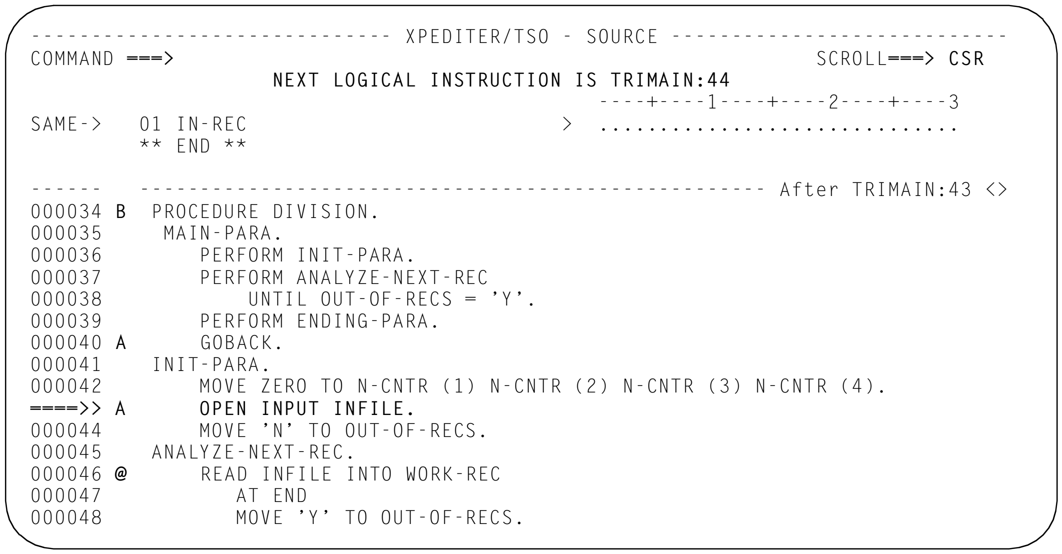
Figure 5-6 shows the two after breakpoints set on statement 43 and 46.The at sign (@) on statement 46 indicates that the before breakpoint was also placed on the same line as the after breakpoint.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                          SCROLL ===> CSR
PROGRAM: TRIRPT      MODULE: TRIMAIN   COMP DATE: 07/28/1997  COMP TIME: 14:41:59



------   -----------------------------------------------------------------------
000042        MOVE ZERO TO N-CNTR (1) N-CNTR (2) N-CNTR (3) N-CNTR (4).
000043 A      OPEN INPUT INFILE.
000044        MOVE 'N' TO OUT-OF-RECS.
000045    ANALYZE-NEXT-REC.
000046 @      READ INFILE INTO WORK-REC
000047           AT END
000048           MOVE 'Y' TO OUT-OF-RECS.
```

Figure 5-6. Both Before and After Breakpoints Set on Line 46

When you press **PF12** (GO), the execution arrow and active breakpoint field shown in Figure 5-7 on page 5-7 indicate that the program is paused after statement 43, the first breakpoint reached during execution.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                    SCROLL===> CSR
                    NEXT LOGICAL INSTRUCTION IS TRIMAIN:44
                                                   ----+----1----+----2----+----3
SAME->   01 IN-REC                          >  .............................
         ** END **

------   --------------------------------------------------- After TRIMAIN:43 <>
000034 B  PROCEDURE DIVISION.
000035     MAIN-PARA.
000036         PERFORM INIT-PARA.
000037         PERFORM ANALYZE-NEXT-REC
000038             UNTIL OUT-OF-RECS = 'Y'.
000039         PERFORM ENDING-PARA.
000040 A       GOBACK.
000041    INIT-PARA.
000042         MOVE ZERO TO N-CNTR (1) N-CNTR (2) N-CNTR (3) N-CNTR (4).
====>> A       OPEN INPUT INFILE.
000044         MOVE 'N' TO OUT-OF-RECS.
000045    ANALYZE-NEXT-REC.
000046 @       READ INFILE INTO WORK-REC
000047            AT END
000048            MOVE 'Y' TO OUT-OF-RECS.
```

Figure 5-7. Result of Entering the GO Command

At this point, you could do any of the following:

- Set additional breakpoints
- Insert XPEDITER/TSO commands into the program
- Enter **GO** to resume execution
- Enter **GO 1** to step through the execution line-by-line
- Enter **EXIT** to exit from the debugging session

For this example, enter the RETEST command to obtain a "fresh" copy of the program.

Setting Conditional Breakpoints

The GO *n*, COUNT, WHEN, PAUSE, and TRACE commands are used to set breakpoints when a specified condition occurs.

Using the GO n Command

Using the GO command without any arguments resumes execution until the next breakpoint is reached, XPEDITER/TSO intercepts a program abend, or the program completes execution. The GO command can also conditionally execute a specified number of statements, paragraphs, or programs if an integer argument is entered with the command. It can also trace each one if the TRACE parameter is entered with the command. For example, entering

```
GO 5
```

executes five *statements* before pausing. Entering

```
GO 5 TRACE
```

executes five *statements* and traces each one, and entering

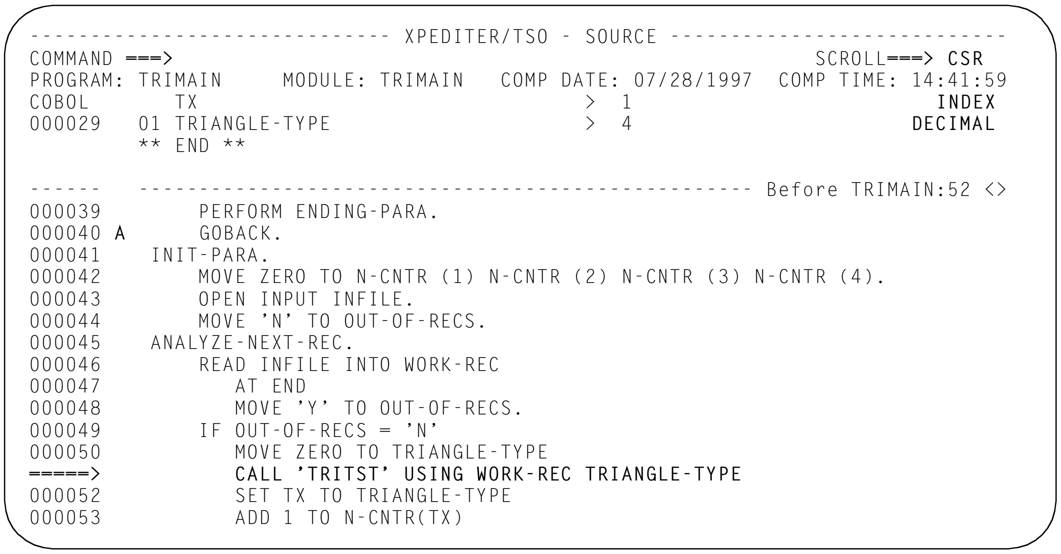
```
GO 5 PARAGRAPH
```

executes five *paragraphs* before pausing.

You can single-step through the code to understand the effect of executing each statement by using the GO 1 command or pressing PF9.The GO 1 command stops at paragraph and section names, as well as at statements that contain the IF construct or any executable verbs. For this example, stop at statement 51.

When a GO 1 command is issued from any statement that transfers control to another module (for example, CALL, GOBACK, EXIT PROGRAM), execution pauses when control returns to the current module, unless a breakpoint is encountered within the called module.

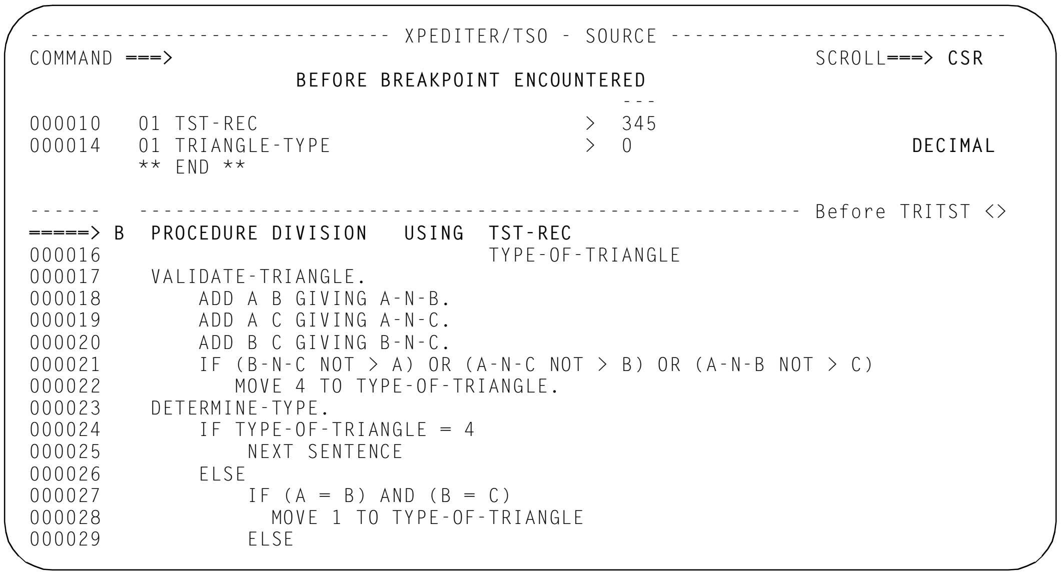
Figure 5-8 shows the result of entering the GO 1 command when execution was paused at the CALL to TRITST at statement 51 and no breakpoints are set in TRITST.



```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                     SCROLL===> CSR
PROGRAM: TRIMAIN     MODULE: TRIMAIN   COMP DATE: 07/28/1997  COMP TIME: 14:41:59
COBOL       TX                          >  1                              INDEX
000029   01 TRIANGLE-TYPE               >  4                            DECIMAL
         ** END **

------   -------------------------------------------------- Before TRIMAIN:52 <>
000039        PERFORM ENDING-PARA.
000040 A      GOBACK.
000041    INIT-PARA.
000042        MOVE ZERO TO N-CNTR (1) N-CNTR (2) N-CNTR (3) N-CNTR (4).
000043        OPEN INPUT INFILE.
000044        MOVE 'N' TO OUT-OF-RECS.
000045    ANALYZE-NEXT-REC.
000046        READ INFILE INTO WORK-REC
000047           AT END
000048           MOVE 'Y' TO OUT-OF-RECS.
000049        IF OUT-OF-RECS = 'N'
000050           MOVE ZERO TO TRIANGLE-TYPE
=====>           CALL 'TRITST' USING WORK-REC TRIANGLE-TYPE
000052           SET TX TO TRIANGLE-TYPE
000053           ADD 1 TO N-CNTR(TX)
```

Figure 5-8. Single Stepping Through Code Within the Current Module

Figure 5-9 demonstrates the result when execution was paused at the CALL to TRITST at statement 51, with a before module breakpoint set at the beginning of TRITST.Then, when you press **PF9** or enter **GO 1**, execution starts in the TRITST module because a BEFORE TRITST: breakpoint was set.



```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                     SCROLL===> CSR
                     BEFORE BREAKPOINT ENCOUNTERED
                                                   ---
000010   01 TST-REC                     >  345
000014   01 TRIANGLE-TYPE               >  0                            DECIMAL
         ** END **

------   ------------------------------------------------------ Before TRITST <>
=====> B  PROCEDURE DIVISION   USING  TST-REC
000016                                TYPE-OF-TRIANGLE
000017    VALIDATE-TRIANGLE.
000018        ADD A B GIVING A-N-B.
000019        ADD A C GIVING A-N-C.
000020        ADD B C GIVING B-N-C.
000021        IF (B-N-C NOT > A) OR (A-N-C NOT > B) OR (A-N-B NOT > C)
000022           MOVE 4 TO TYPE-OF-TRIANGLE.
000023    DETERMINE-TYPE.
000024        IF TYPE-OF-TRIANGLE = 4
000025            NEXT SENTENCE
000026        ELSE
000027            IF (A = B) AND (B = C)
000028              MOVE 1 TO TYPE-OF-TRIANGLE
000029            ELSE
```

Figure 5-9. Stepping Into a Called Module Using the GO 1 Command

To bring a new copy of your source into the display, enter **RETEST**.

Using the COUNT MAX Command

The COUNT command is used to monitor execution coverage by maintaining statement execution counts. When the MAX keyword is used with the COUNT command, a conditional breakpoint with a count limit is set. When the limit is reached, program execution pauses and the message SPECIFIED EXECUTION MAX HAS BEEN REACHED is displayed. When the COUNT command is issued, a 7-digit counter appears in columns 74 through 80.

```
=====>     ANALYZE-NEXT-REC. 0000003
```

The counter field can be typed over to set and reset the limit. For example, you can type over the counter with a higher limit and press **Enter** to raise the preset maximum limit, or you can zero out the counter and press **Enter** to reset the limit.

Using the WHEN Command

The WHEN command lets you stop execution when a program variable changes value or when a specified event takes place. XPEDITER/TSO checks the condition after every statement in the current module and pauses if the condition is met. The WHEN command can be used with the following arguments:

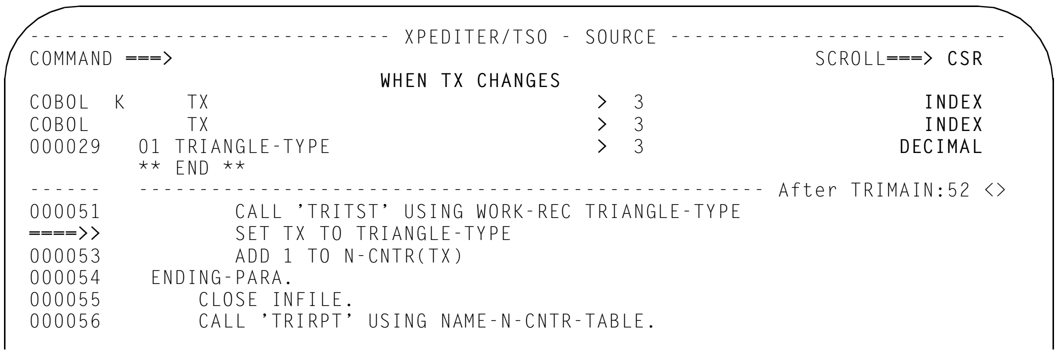
Variable-name	Suspends execution when a statement altering the value of the variable is executed. The variable content can be monitored by opening a Keep window and displaying the variable content.
Condition	Suspends execution when the specified condition is met. You can enter a relational condition using expressions such as the following:

```
WHEN WORK-REC = '345'
WHEN TOTAL-SUM > 50000
WHEN WS-TRAN-KEY = HIGH-VALUES
WHEN OUT-OF-RECS CHANGES
```

For example, enter the following WHEN command to conditionally pause when the index TX is changed, and enter the KEEP command on index TX to monitor the change:

```
WHEN TX;KEEP TX
```

Figure 5-10 shows the result of resuming execution by pressing **PF12** (GO). Note that the automatic keep function also keeps the value of TX and TRIANGLE-TYPE. However, these values will disappear as the current line changes, but the KEEP command will continuously monitor the value of TX. The DELETE WHEN command can be used to remove the when condition.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                     SCROLL===> CSR
                               WHEN TX CHANGES
COBOL  K      TX                          >  3                           INDEX
COBOL         TX                          >  3                           INDEX
000029   01 TRIANGLE-TYPE                 >  3                         DECIMAL
         ** END **
------   --------------------------------------------------- After TRIMAIN:52 <>
000051            CALL 'TRITST' USING WORK-REC TRIANGLE-TYPE
====>>            SET TX TO TRIANGLE-TYPE
000053            ADD 1 TO N-CNTR(TX)
000054     ENDING-PARA.
000055         CLOSE INFILE.
000056         CALL 'TRIRPT' USING NAME-N-CNTR-TABLE.
```

Figure 5-10. Reaching the When Breakpoint

Using the Inserted PAUSE Command

The PAUSE command can be dynamically inserted in the COBOL source code to set a breakpoint following the execution of a statement. You can specify a condition in which the pause breakpoint is to occur, by also inserting the IF...ELSE construct associated with it.

The PAUSE command **must** be used in conjunction with the INSERT command. For information on the usage convention of the INSERT command and the effect of inserting a PAUSE command, refer to "Inserting Statements" on page 5-36.

Using the TRACE Command

The TRACE command is used to monitor the execution of specified statements or paragraphs in your program. The specified statements or paragraph names are highlighted as they are executed until a breakpoint is reached, an abend is intercepted by XPEDITER/TSO, a terminal I/O is issued, or a keyboard interrupt is detected.

When the TRACE command is used with the MAX keyword, the tracing function pauses when the number of executions reaches the preset limit. The default maximum limit is 25.

For example, if you enter the following command for a program that does not have any breakpoints set and press **PF12** (GO) to resume execution:

```
TRACE ALL STATEMENTS
```

the program will pause following the execution of 25 statements and display the message `25 TRACE BREAKPOINTS HAVE BEEN EXECUTED`.

In order to override the default maximum limit, you must enter the MAX keyword with an integer other than 25 as an argument with the TRACE command.

The tracing speed can be controlled by using the SET DELAY command prior to entering the TRACE command. For instance, entering the following command will slow down the execution speed to one second:

```
SET DELAY 1
```

You can interrupt tracing and suspend execution by using the Attention key. While tracing is in progress, the keyboard is unlocked and, depending upon your terminal type and network configuration, you may be able to use other keys to stop tracing. To end a TRACE command, use DELETE TRACE.

Inspecting Program Data

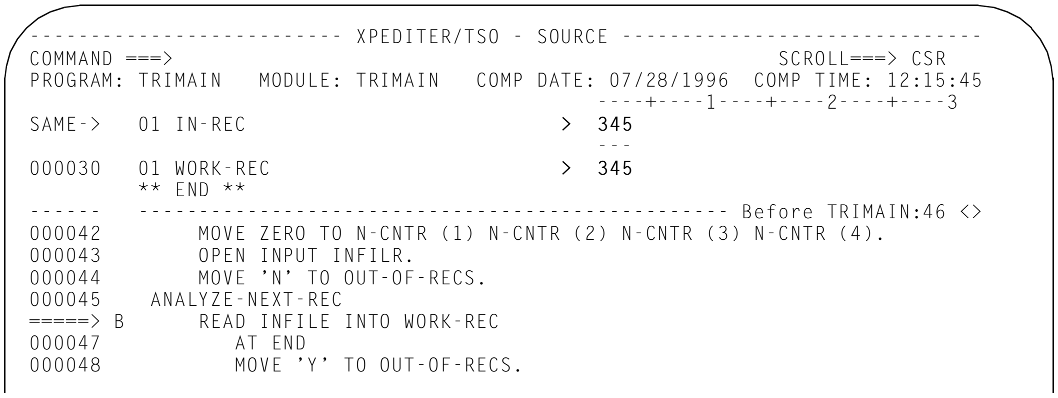
XPEDITER/TSO lets you view the contents of variables defined in your program and the data that is passed to the working storage and the linkage. The data is formatted by the data type defined in your program.

Described in this section are the Automatic Keep function and the XPEDITER/TSO commands KEEP, PEEK, MOVE, MEMORY, and GPREGS.

When you have XPEDITER for DB2 Extension and File-AID for DB2 installed, you can also browse and edit DB2 table data during an XPEDITER/TSO session. Refer to "Using XPEDITER for DB2 Extension" on page 5-40 for more information.

Displaying and Modifying Program Variables

XPEDITER/TSO automatically displays the values of data items referenced by the current execution statement whenever execution halts. These values are displayed in a Keep window at the top of the source display as shown in Figure 5-11 on page 5-11. Each time the program halts, a new set of variables and their values are displayed.



```
-------------------------- XPEDITER/TSO - SOURCE -----------------------------
COMMAND ===>                                                   SCROLL===> CSR
PROGRAM: TRIMAIN   MODULE: TRIMAIN   COMP DATE: 07/28/1996  COMP TIME: 12:15:45
                                                           ----+----1----+----2----+----3
SAME->   01 IN-REC                           >  345
                                                ---
000030   01 WORK-REC                         >  345
         ** END **
------   ------------------------------------------------ Before TRIMAIN:46 <>
000042        MOVE ZERO TO N-CNTR (1) N-CNTR (2) N-CNTR (3) N-CNTR (4).
000043        OPEN INPUT INFILR.
000044        MOVE 'N' TO OUT-OF-RECS.
000045    ANALYZE-NEXT-REC
=====> B      READ INFILE INTO WORK-REC
000047           AT END
000048           MOVE 'Y' TO OUT-OF-RECS.
```

Figure 5-11. Keep Window Displaying Automatic Keeps

If a variable of interest is not automatically kept, you can use an explicit KEEP command or the PEEK command to temporarily display the contents of the variable.

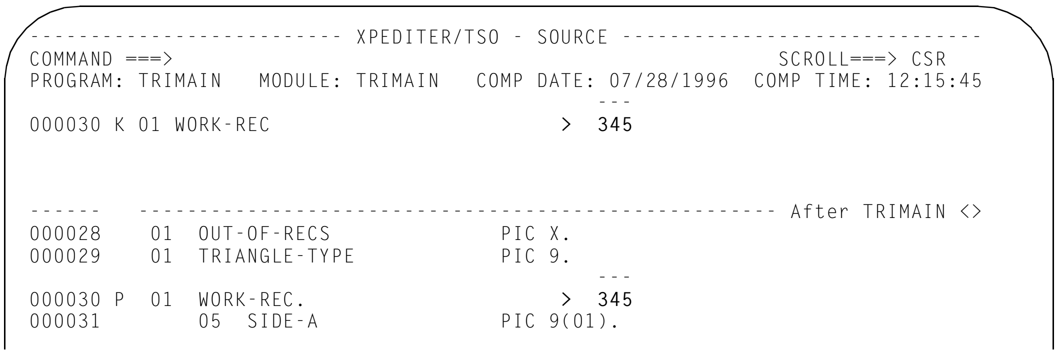
The KEEP command is used when you want to continuously display the contents of a variable. When the KEEP command is entered, XPEDITER/TSO continuously displays and updates the data values in the Keep window until you delete the keep. XPEDITER inserts a K in column 9 of the Keep window to differentiate between the explicitly kept items and the automatically kept items.

Note: You have the option of creating a separate window called the Automatic Keep window located at the bottom of the source display to hold the automatically kept items. To do this, use the SET AUTOKEEP *n* command. Refer to the SET command in the *XPEDITER/TSO and XPEDITER/IMS Reference Manual* for additional information.

The PEEK command is used when you want to temporarily display the contents of a variable. When the PEEK command is entered, XPEDITER/TSO scrolls to the DATA DIVISION statement, displays the data values in a window on the right side of the screen, and inserts a P in column 9 of the source. Some of the statement will be overlaid. When you resume execution, the window is removed from the screen. Examples of the Keep and Peek windows are shown in Figure 5-12.

Notes::

1. Use the LOCATE * command (PF6), to return to the location where execution is paused.
2. Use the DELETE command to remove a Keep or Peek display.
3. Use the SET AUTOKEEP OFF command to turn off the Automatic Keep function.



```
-------------------------- XPEDITER/TSO - SOURCE -----------------------------
COMMAND ===>                                                   SCROLL===> CSR
PROGRAM: TRIMAIN   MODULE: TRIMAIN   COMP DATE: 07/28/1996  COMP TIME: 12:15:45
                                                ---
000030 K 01 WORK-REC                         >  345



------   ---------------------------------------------------- After TRIMAIN <>
000028    01  OUT-OF-RECS              PIC X.
000029    01  TRIANGLE-TYPE            PIC 9.
                                                ---
000030 P  01  WORK-REC.                      >  345
000031        05  SIDE-A               PIC 9(01).
```

Figure 5-12. Keep and Peek Windows

Automatic Keep, Keep, and Peek windows display data in the same format: the statement number where the variable is declared, the level number, the data name, the current value, and the data type.

The level number appears only if the variable is declared in the source as a structure with level numbers. Data values are displayed according to scaling and precision attributes. Alphabetic items are displayed as characters, and numeric items are displayed as DECIMAL, PACKED decimal (COMP-3), HALFWORD (COMP), FULLWORD (COMP), INDEX, or FLOAT (COMP-1, COMP-2), depending on the internal representation. Tables are shown by rows and columns, rather than in a linear fashion.

The Keep window is both scrollable and adjustable in size. The window becomes scrollable when the data exceeds the size of the window. Scrolling is cursor sensitive; that is, the cursor must be in the Keep window for vertical scrolling to take place. Move the cursor to the window and use the PF7 (UP) and PF8 (DOWN) keys to scroll the data vertically. Use the PF22 (DRIGHT) and PF23 (DLEFT) keys to scroll the Keep and Peek windows horizontally. To ensure that the cursor remains in the Keep window while scrolling, put the cursor on the segmented execution status line before using your scroll keys.

You can set the size of the Source, the Keep window, and the Automatic Keep window. The automatically kept items can, by default, be displayed in the Keep window at the top of the screen or in a separate Automatic Keep window at the bottom of the screen. Refer to the SET command in the *XPEDITER/TSO and XPEDITER/IMS Reference Manual* for additional information.

The contents of variables displayed in a window can be changed by using the MOVE command or by typing over them (implicit MOVE). The OCCURS field can also be typed over to browse through the table by each entry. A relative subscript can be appended to increment or decrement the occurrences.

Using the KEEP Command

The KEEP command lets you continuously view the contents of program variables in a window opened at the top of the source display. You can enter the KEEP primary command with the name of the variable or you can enter the K line command either in the Procedure Division (where the variable is referenced) or in the Data Division (where the variable is defined). If the data exceeds 30 bytes, the field becomes scrollable, and it is indicated by the MORE-> sign in the statement number area.Tables are formatted by each dimension when the KEEP command is entered at the elementary level. The OCCURS field can be typed over to browse through the table by each entry. Also, a relative subscript can be appended to increment or decrement the occurrences. If the data displayed is a character data type, a column template is displayed to show the length. The length of a numeric item, however, is a function of the internal format.The alphabetic items are displayed as characters and the numeric items are displayed as DECIMAL, PACKED decimal (COMP-3), HALFWORD (COMP), FULLWORD (COMP), INDEX, or FLOAT (COMP-1, COMP-2), depending on the internal representation. Tables will be shown by row and column, rather than in a linear fashion.

The KEEP command has the following format:

KEEP (K line command) Keeps the value of a variable.

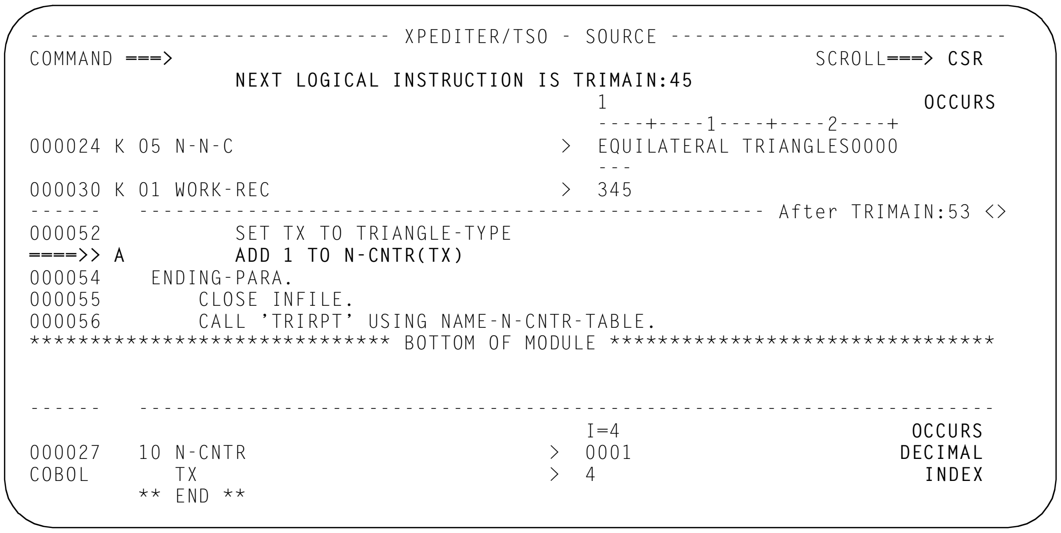
KEEPE (KE line command) Keeps the values of the elementary items that are part of a group item.

KEEPH (KH line command) Keeps the hexadecimal values of a variable.

Use the DELETE command (DELETE KEEP or D line command) to remove the display resulting from the KEEP commands.

Examples for using the KEEP command are given below. Note that an AFTER 53 breakpoint was entered in the TRIMAIN program before execution was resumed.

Figure 5-13 on page 5-13 shows the result of entering the KEEP command for the variables WORK-REC and N-N-C. For this example, the automatic Keep window was moved to the bottom of the display with the SET AUTOKEEP 5 command so that all of the explicitly kept items will show in the Keep window without scrolling.

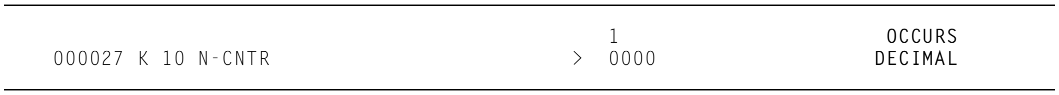


```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                     SCROLL===> CSR
                 NEXT LOGICAL INSTRUCTION IS TRIMAIN:45
                                                 1                       OCCURS
                                                 ----+----1----+----2----+
000024 K 05 N-N-C                              > EQUILATERAL TRIANGLES0000
                                                 ---
000030 K 01 WORK-REC                           > 345
------  --------------------------------------------------- After TRIMAIN:53 <>
000052           SET TX TO TRIANGLE-TYPE
====>> A         ADD 1 TO N-CNTR(TX)
000054    ENDING-PARA.
000055        CLOSE INFILE.
000056        CALL 'TRIRPT' USING NAME-N-CNTR-TABLE.
***************************** BOTTOM OF MODULE *******************************



------  -----------------------------------------------------------------------
                                                I=4                      OCCURS
000027   10 N-CNTR                            > 0001                    DECIMAL
COBOL       TX                                > 4                         INDEX
         ** END **
```

Figure 5-13. Result of Entering the KEEP Command for WORK-REC and N-N-C

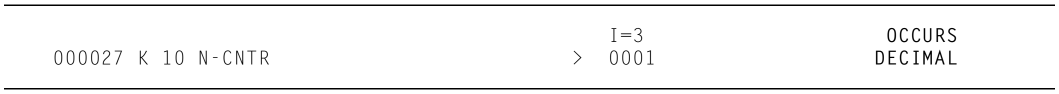
When a table element is displayed, XPEDITER/TSO inserts an OCCURS field as shown in Figure 5-13. The occurrence number defaults to one. If you need to display the currently referenced data in the table as the subscript or index is changed, enter the KEEP command on a data item referenced by the subscript or the index.

For instance, a KEEP on N-CNTR displays the data value in the window defaulting to the first occurrence (Figure 5-14), whereas a KEEP on N-CNTR(TX) displays the currently referenced data (Figure 5-15).



```
                                                 1                     OCCURS
  000027 K 10 N-CNTR                           > 0000                 DECIMAL
```

Figure 5-14. Result of Entering the KEEP Command on a Table Element N-CNTR

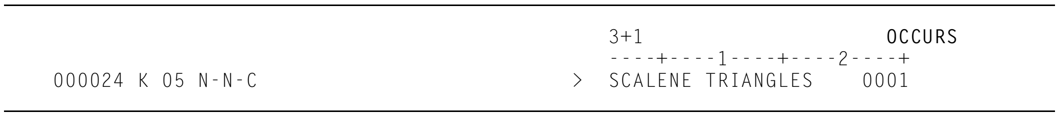


```
                                                 I=3                   OCCURS
  000027 K 10 N-CNTR                           > 0001                 DECIMAL
```

Figure 5-15. Result of Entering the KEEP Command on a Table Element N-CNTR(TX)

The value in the OCCURS field can be typed over with a valid subscript to display the subsequent table entry items. Also, a relative subscript can be appended to the OCCURS field in order to browse through the table elements each time you press **Enter**.

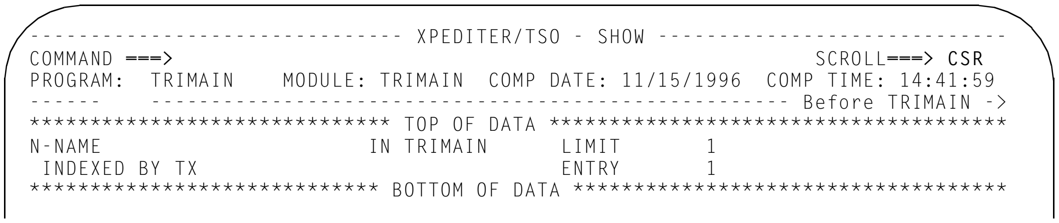
As shown in Figure 5-16, add a signed integer (for example, +1, -2) to increment or decrement the subscript by a specified interval.



```
                                                 3+1                   OCCURS
                                                 ----+----1----+----2----+
  000024 K 05 N-N-C                            > SCALENE TRIANGLES    0001
```

Figure 5-16. Browsing Through the Table Elements

The subscript and index boundaries are automatically checked when you are browsing through the table. The boundary limit can be displayed by entering the SHOW INDEX command. Figure 5-17 on page 5-14 shows the boundary limits for N-N-C.



```
------------------------------ XPEDITER/TSO - SHOW ----------------------------
COMMAND ===>                                                     SCROLL===> CSR
PROGRAM:  TRIMAIN    MODULE: TRIMAIN  COMP DATE: 11/15/1996  COMP TIME: 14:41:59
------    ---------------------------------------------------- Before TRIMAIN ->
*************************** TOP OF DATA ************************************
N-NAME                       IN TRIMAIN      LIMIT       1
 INDEXED BY TX                               ENTRY       1
**************************** BOTTOM OF DATA *********************************
```

Figure 5-17. Result of the SHOW INDEX Command

Using the PEEK Command

You can enter the PEEK primary command with the name of the variable or you can enter the P line command either in the Procedure Division (where the variable is referenced) or in the Data Division (where the variable is defined).

There are three forms of the PEEK command, each corresponding to a data display format. The three forms are:

PEEK (P line command) Displays the value of a variable.

PEEKE (PE line command) Displays the values of the elementary items that are part of a group item.

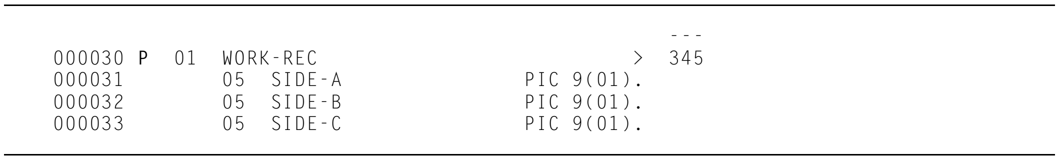
PEEKH (PH line command) Displays the hexadecimal values of a variable.

Examples of the commands and the resulting formats are described below.

Use the DELETE primary command or line commands to remove the display resulting from the PEEK commands.

When the PEEK command is issued, the value of the variable is displayed in character or numeric format.Tables are formatted by each dimension when the PEEK command is entered at the elementary level. If the data displayed is a character data type, a column template is displayed to show the length. The length of a numeric item, however, is a function of the internal format.

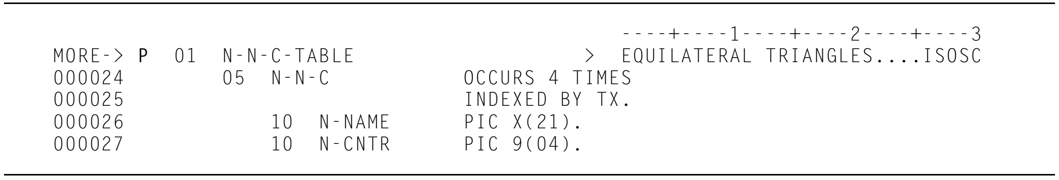
Figure 5-18 shows the result of entering the PEEK command on WORK-REC in the TRIMAIN program.



```
                                                   ---
000030 P   01  WORK-REC                          > 345
000031         05  SIDE-A               PIC 9(01).
000032         05  SIDE-B               PIC 9(01).
000033         05  SIDE-C               PIC 9(01).
```

Figure 5-18. Result of Entering PEEK WORK-REC

Figure 5-19 shows the result of entering the PEEK command on N-N-C-TABLE in the TRIMAIN program.Note the MORE-> sign in the statement number area and the column template.



```
                                               ----+----1----+----2----+----3
MORE-> P   01  N-N-C-TABLE                   > EQUILATERAL TRIANGLES....ISOSC
000024         05  N-N-C          OCCURS 4 TIMES
000025                            INDEXED BY TX.
000026             10  N-NAME     PIC X(21).
000027             10  N-CNTR     PIC 9(04).
```

Figure 5-19. Result of Entering PEEK N-N-C-TABLE (Shows MORE>Sign and Column Template)

Figure 5-20 on page 5-15 shows the result of entering the PEEK command on N-N-C in the TRIMAIN program. Note the appearance of the OCCURS field. The OCCURS field can be incremented or decremented by typing over it.

```
                                                  1                      OCCURS
                                                  ----+----1----+----2----+
000024 P      05  N-N-C                     >  EQUILATERAL TRIANGLES....
000025                            INDEXED BY TX.
000026            10  N-NAME      PIC X(21).
000027            10  N-CNTR      PIC 9(04).
```

Figure 5-20. Result of Entering PEEK N-N-C (Shows OCCURS Field and Column Template)

Figure 5-21 shows the result of entering the PEEK command on CHECK-SUM in a program that contains counters and sums. Note that the numeric value is shown in PACKED decimal format.

```
000252    01  WS-SUMS.
000254 P      05  CHECK-SUM                 >  +2001474.01           PACKED
```

Figure 5-21. Result of Entering PEEK CHECK-SUM (Shows PACKED Decimal Format)

Figure 5-22 shows the result of entering the PEEK command on CHAR-PTR in a program that contains halfwords. Note that the numeric value is shown in HALFWORD format.

```
000260    01  WS-POINTERS
000262 P      05  CHAR-PTR                  >  +093                HALFWORD
```

Figure 5-22. Result of Entering PEEK CHAR-PTR (Shows HALFWORD Format)

When the PEEKE command is entered for a group level data name containing elementary data items, the values for each elementary item are displayed, as shown in Figure 5-23, where PEEKE was entered on WORK-REC in the TRIMAIN program.

```
000030    01  WORK-REC
000031 P      05  SIDE-A            PIC 9 >  3                     DECIMAL
000032 P      05  SIDE-B            PIC 9 >  4                     DECIMAL
000033 P      05  SIDE-C            PIC 9 >  5                     DECIMAL
```

Figure 5-23. Result of Entering PEEKE Command on the Group Level Data Item WORK-REC (Shows DECIMAL Format)

The PEEKH command displays the value of the variable in the hexadecimal format. Figure 5-24 shows the result of entering the PEEKH command on the 05 data item SIDE-A for the 01 data item WORK-REC in the TRIMAIN program.

```
000031 P      05  SIDE-A            PIC 9 >  3                      DECIMAL
                                             F
                                             3
000032        05  SIDE-B            PIC 9(01).
000033        05  SIDE-C            PIC 9(01).
```

Figure 5-24. Result of Entering PEEKH Command on the Group Level Data Item SIDE-A (Shows HEXADECIMAL Format)

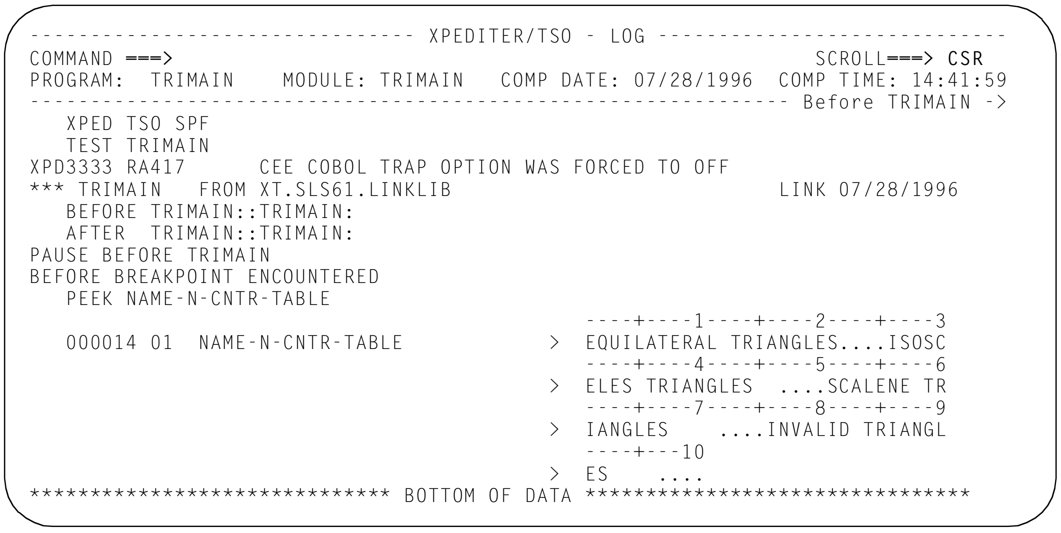
The Log Entries for Automatic KEEP, KEEP, and PEEK

Every time a PEEK or KEEP command is entered, or an Automatic Keep displays data in the Keep window, the command and the value of the data name are entered in the log. The format of the log entry for both the PEEK and KEEP commands is similar.

To reduce the size of the log, you can turn off the log entries of the KEEP and PEEK commands and Automatic Keeps by using the SET LOG KEEP OFF, SET LOG PEEK OFF, and SET LOG AUTOKEEP OFF commands.

Since a displayed field is not scrollable in the log, large alphanumeric items wrap around to the next line. The number of characters of data per line for an alphanumeric display is determined by the value specified on the SET LOGSIZE command.The record length for the log file can be set to either 80 or 133. If the LOGSIZE is 80, an alphanumeric display wraps around after 30 characters per line. If the LOGSIZE is 133, the display wraps around after 80 characters per line.

Figure 5-25 shows a log entry for the PEEK command entered for a table. The LOGSIZE is set to 80. In this example, the values in the displayed table wrap around every 30 bytes until the entire table of 100 bytes is displayed.



```
-------------------------------- XPEDITER/TSO - LOG ----------------------------
COMMAND ===>                                                   SCROLL===> CSR
PROGRAM:  TRIMAIN    MODULE: TRIMAIN   COMP DATE: 07/28/1996  COMP TIME: 14:41:59
-------------------------------------------------------------- Before TRIMAIN ->
   XPED TSO SPF
   TEST TRIMAIN
XPD3333 RA417      CEE COBOL TRAP OPTION WAS FORCED TO OFF
*** TRIMAIN   FROM XT.SLS61.LINKLIB                           LINK 07/28/1996
   BEFORE TRIMAIN::TRIMAIN:
   AFTER  TRIMAIN::TRIMAIN:
PAUSE BEFORE TRIMAIN
BEFORE BREAKPOINT ENCOUNTERED
   PEEK NAME-N-CNTR-TABLE
                                        ----+----1----+----2----+----3
   000014 01  NAME-N-CNTR-TABLE       >  EQUILATERAL TRIANGLES....ISOSC
                                        ----+----4----+----5----+----6
                                      >  ELES TRIANGLES  ....SCALENE TR
                                        ----+----7----+----8----+----9
                                      >  IANGLES    ....INVALID TRIANGL
                                        ----+---10
                                      >  ES    ....
****************************** BOTTOM OF DATA *******************************
```

Figure 5-25. The Log Entry Following Execution of a PEEK Command

Unless SET HEXMODE is specified as ON, the log entry of a displayed field containing nonrepresentable characters includes the symbol used to represent them—either periods or the character designated by the SET NONDISP command.

When the SET HEXMODE command is on, nonrepresentable characters are displayed in hexadecimal format in the log. SET HEXMODE ON ensures that sufficient information is provided for a variable containing unprintable characters. All invalid numeric data (e.g., uninitialized packed data) is represented by question marks (?).

When an after, before, trace, when, or GO 1 breakpoint is encountered and the value of one or more of the kept variables changes, all kept variables are entered in the log. The variables whose values have changed are listed first, followed by the data names that have not changed. If none of the values of the kept variables change, a log entry is not made.

Using the MOVE Command

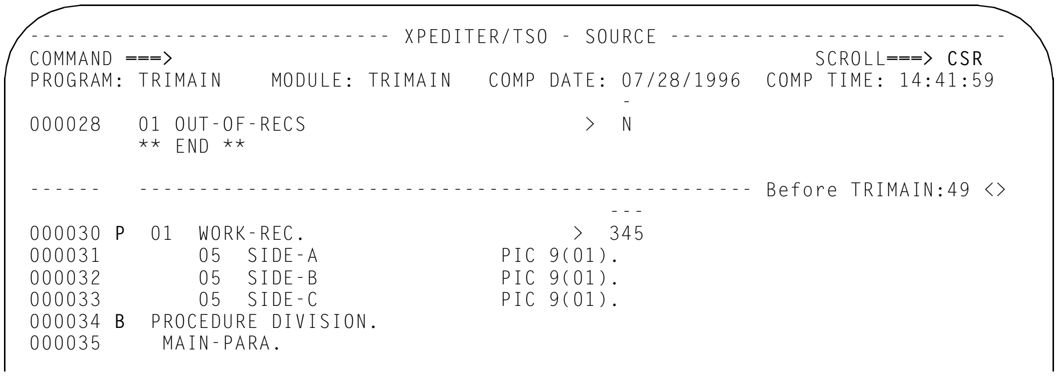
You can change the contents of program variables at any time using the MOVE command. MOVE lets you move either a data name or a literal into another data name.

There are three ways to enter the MOVE command:

1. By typing over a displayed or kept field with a new value, causing an implicit move.
2. Directly as a primary command.
3. In conjunction with the INSERT command.

The log entry for all three MOVE command formats is the same.

The following figures show examples of entering the MOVE command in different ways to produce the same result. The examples will use Figure 5-26 on page 5-17 in which the variable WORK-REC is displayed.

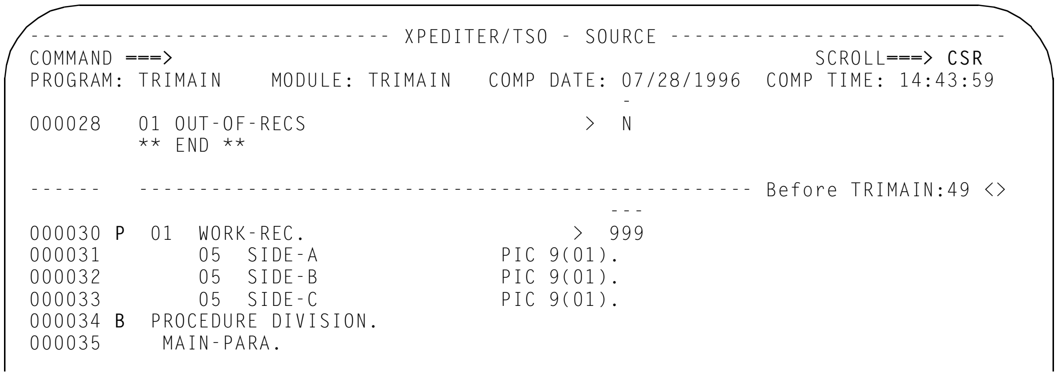


```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                      SCROLL===> CSR
PROGRAM: TRIMAIN    MODULE: TRIMAIN   COMP DATE: 07/28/1996  COMP TIME: 14:41:59
                                                 -
000028   01 OUT-OF-RECS                        >  N
         ** END **

------   -------------------------------------------------- Before TRIMAIN:49 <>
                                                ---
000030 P  01  WORK-REC.                        >  345
000031         05  SIDE-A               PIC 9(01).
000032         05  SIDE-B               PIC 9(01).
000033         05  SIDE-C               PIC 9(01).
000034 B  PROCEDURE DIVISION.
000035     MAIN-PARA.
```

Figure 5-26. Displaying Variable WORK-REC Prior to Typing Over Value

Example 1 — Typing Over Value in Variable Field to Cause an Implicit Move:

In Figure 5-27, the displayed value (345) for the variable WORK-REC is typed over with the new value (999), causing an implicit move.

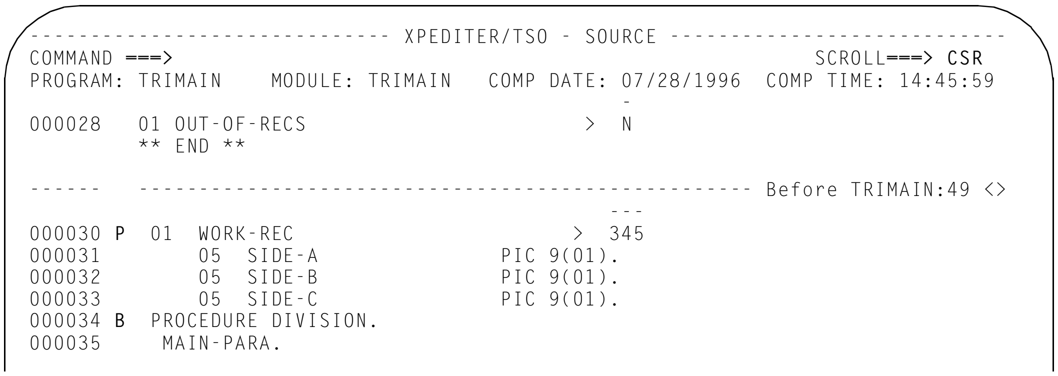


```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                      SCROLL===> CSR
PROGRAM: TRIMAIN    MODULE: TRIMAIN   COMP DATE: 07/28/1996  COMP TIME: 14:43:59
                                                 -
000028   01 OUT-OF-RECS                        >  N
         ** END **

------   -------------------------------------------------- Before TRIMAIN:49 <>
                                                ---
000030 P  01  WORK-REC.                        >  999
000031         05  SIDE-A               PIC 9(01).
000032         05  SIDE-B               PIC 9(01).
000033         05  SIDE-C               PIC 9(01).
000034 B  PROCEDURE DIVISION.
000035     MAIN-PARA.
```

Figure 5-27. Typing Over Value for Variable WORK-REC

Example 2 — Entering MOVE as a Primary Command:

If you enter **MOVE '345' TO WORK-REC** in the primary command line, the result is an explicit move.As shown in Figure 5-28, the literal value (345) is moved to the data field for WORK-REC and the value of WORK-REC is changed from 999 to 345.

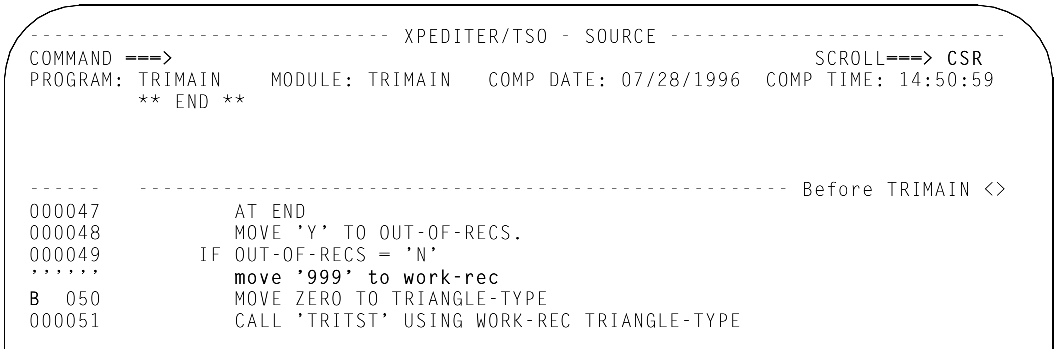


```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                      SCROLL===> CSR
PROGRAM: TRIMAIN    MODULE: TRIMAIN   COMP DATE: 07/28/1996  COMP TIME: 14:45:59
                                                 -
000028   01 OUT-OF-RECS                        >  N
         ** END **

------   -------------------------------------------------- Before TRIMAIN:49 <>
                                                ---
000030 P  01  WORK-REC                         >  345
000031         05  SIDE-A               PIC 9(01).
000032         05  SIDE-B               PIC 9(01).
000033         05  SIDE-C               PIC 9(01).
000034 B  PROCEDURE DIVISION.
000035     MAIN-PARA.
```

Figure 5-28. An Example of MOVE Entered as a Primary Command

Example 3 — Using the MOVE Command in Conjunction With the INSERT Command:

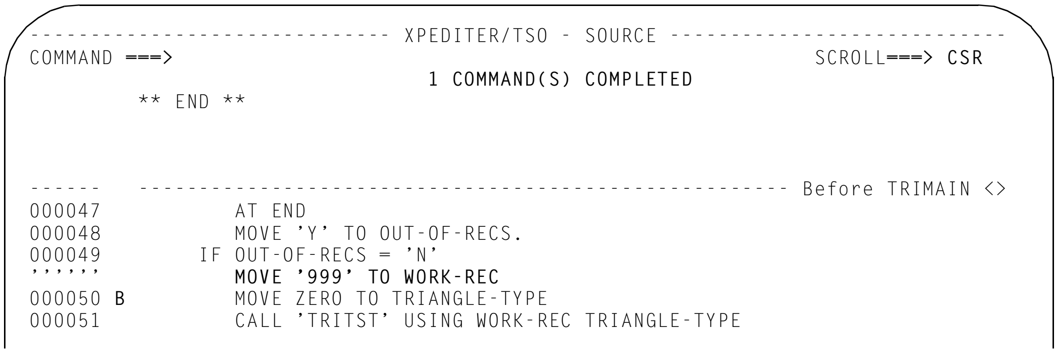
In Figure 5-29 on page 5-18, the I (Insert) line command was entered on line 49 to open up a line on which to insert the MOVE command.A before breakpoint is also being entered on line 50, following the inserted MOVE command.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                    SCROLL===> CSR
PROGRAM: TRIMAIN    MODULE: TRIMAIN   COMP DATE: 07/28/1996  COMP TIME: 14:50:59
         ** END **



------   ------------------------------------------------------ Before TRIMAIN <>
000047           AT END
000048           MOVE 'Y' TO OUT-OF-RECS.
000049        IF OUT-OF-RECS = 'N'
''''''           move '999' to work-rec
B  050           MOVE ZERO TO TRIANGLE-TYPE
000051           CALL 'TRITST' USING WORK-REC TRIANGLE-TYPE
```

Figure 5-29. Inserting a MOVE Command

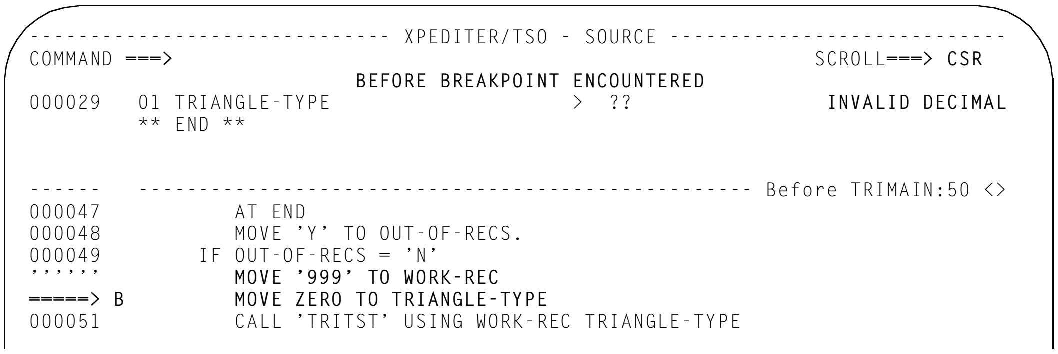
When you press **Enter**, a before breakpoint is indicated for line 50, as shown in Figure 5-30.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                    SCROLL===> CSR
                                1 COMMAND(S) COMPLETED
         ** END **



------   ------------------------------------------------------ Before TRIMAIN <>
000047           AT END
000048           MOVE 'Y' TO OUT-OF-RECS.
000049        IF OUT-OF-RECS = 'N'
''''''           MOVE '999' TO WORK-REC
000050 B         MOVE ZERO TO TRIANGLE-TYPE
000051           CALL 'TRITST' USING WORK-REC TRIANGLE-TYPE
```

Figure 5-30. Before Breakpoint Set on Line 50

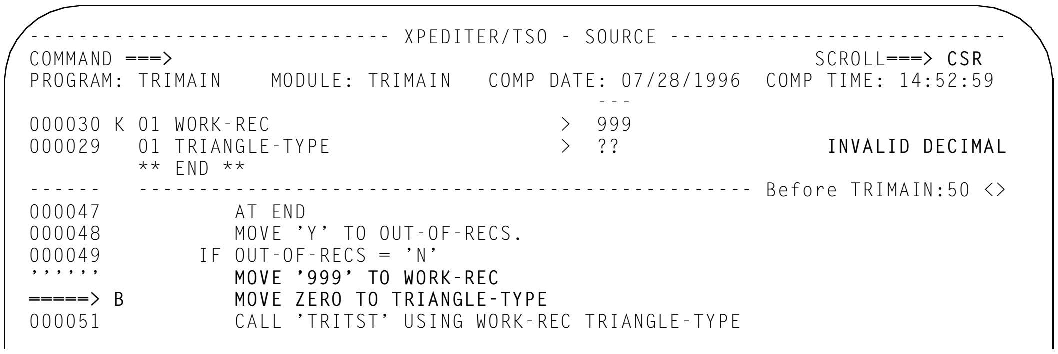
Then, when you press **PF12** (GO), execution is paused following the inserted MOVE command as shown in Figure 5-31.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                    SCROLL===> CSR
                          BEFORE BREAKPOINT ENCOUNTERED
000029   01 TRIANGLE-TYPE                  >  ??               INVALID DECIMAL
         ** END **


------   --------------------------------------------------- Before TRIMAIN:50 <>
000047           AT END
000048           MOVE 'Y' TO OUT-OF-RECS.
000049        IF OUT-OF-RECS = 'N'
''''''           MOVE '999' TO WORK-REC
=====> B         MOVE ZERO TO TRIANGLE-TYPE
000051           CALL 'TRITST' USING WORK-REC TRIANGLE-TYPE
```

Figure 5-31. Source Display After GO is Entered

Enter **KEEP** for WORK-REC. The value indicated on the inserted MOVE command is displayed in the Keep window as shown in Figure 5-32 on page 5-19.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                  SCROLL===> CSR
PROGRAM: TRIMAIN    MODULE: TRIMAIN   COMP DATE: 07/28/1996  COMP TIME: 14:52:59
                                                  ---
000030 K 01 WORK-REC                           >  999
000029   01 TRIANGLE-TYPE                      >  ??                INVALID DECIMAL
         ** END **
------   -------------------------------------------------- Before TRIMAIN:50 <>
000047           AT END
000048           MOVE 'Y' TO OUT-OF-RECS.
000049        IF OUT-OF-RECS = 'N'
''''''           MOVE '999' TO WORK-REC
=====> B         MOVE ZERO TO TRIANGLE-TYPE
000051           CALL 'TRITST' USING WORK-REC TRIANGLE-TYPE
```

Figure 5-32. Result of an Inserted MOVE Command

Displaying and Modifying Memory and Registers

Storage areas and general-purpose registers can be accessed if you wish to debug at the hexadecimal level. The MEMORY command lets you view and modify the storage area starting from a specified location. A full screen memory display is shown in the dump format, and any unprotected areas can be typed over to alter the storage content. The GPREGS command lets you view and modify the registers. A register window is displayed at the bottom of the screen, and you can type over the hexadecimal values to modify the registers. The TOGGLE command permits you to switch back and forth between the storage screen (generated from the MEMORY command) and the listing screen (general-purpose registers opened by using the GPREGS command).

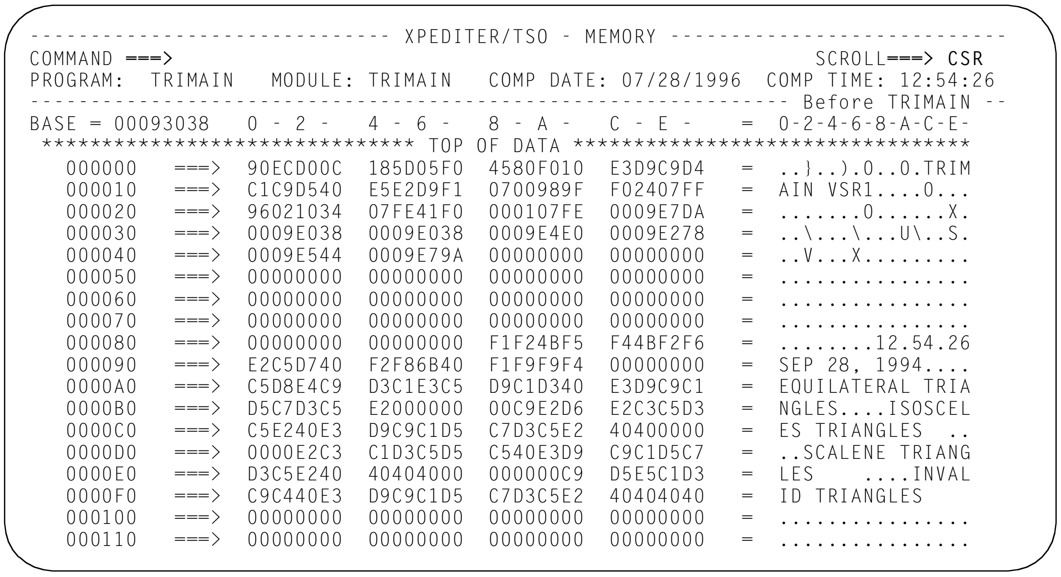
Using the MEMORY Command

The MEMORY command entered without any arguments displays the storage area, starting from the beginning of the currently displayed program.For example, entering the following command from the primary command line of the TRIMAIN program displays the Memory Display screen shown in Figure 5-33:

```
MEMORY
```

The fourth line on the Memory Display screen contains the base address and a column template that starts with zero and continues to hexadecimal F. The offsets below the base address list the displacement from the start of the storage area.

Press **PF3** (END) to return to the Source display screen.

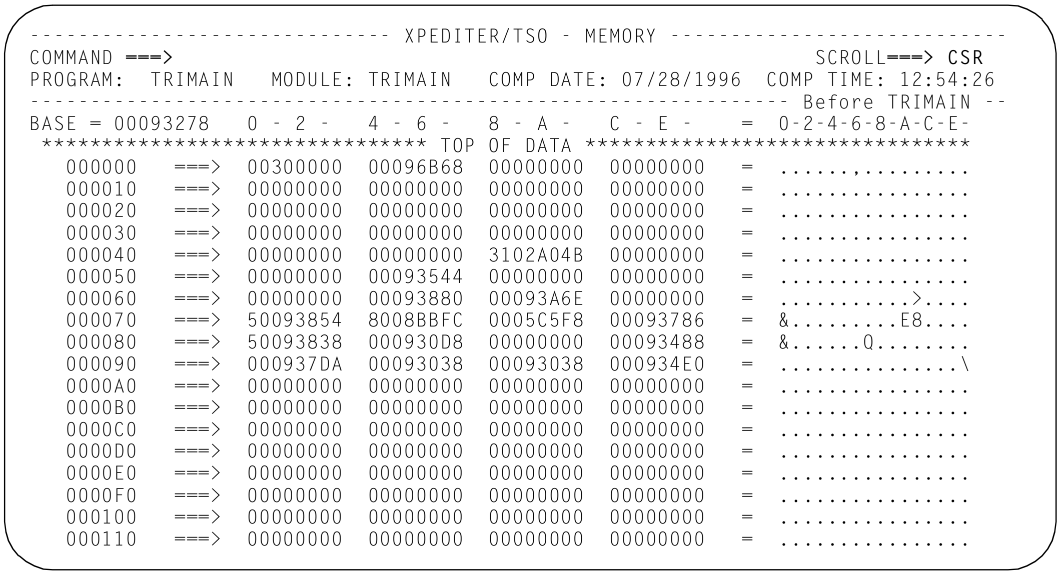


```
------------------------------ XPEDITER/TSO - MEMORY ---------------------------
COMMAND ===>                                                  SCROLL===> CSR
PROGRAM:  TRIMAIN   MODULE: TRIMAIN   COMP DATE: 07/28/1996  COMP TIME: 12:54:26
--------------------------------------------------------------- Before TRIMAIN --
BASE = 00093038   0 - 2 -   4 - 6 -   8 - A -   C - E -    =  0-2-4-6-8-A-C-E-
 ****************************** TOP OF DATA *******************************
   000000   ===>  90ECD00C  185D05F0  4580F010  E3D9C9D4   =  ..}..).0..0.TRIM
   000010   ===>  C1C9D540  E5E2D9F1  0700989F  F02407FF   =  AIN VSR1....0...
   000020   ===>  96021034  07FE41F0  000107FE  0009E7DA   =  .......0......X.
   000030   ===>  0009E038  0009E038  0009E4E0  0009E278   =  ..\...\...U\..S.
   000040   ===>  0009E544  0009E79A  00000000  00000000   =  ..V...X.........
   000050   ===>  00000000  00000000  00000000  00000000   =  ................
   000060   ===>  00000000  00000000  00000000  00000000   =  ................
   000070   ===>  00000000  00000000  00000000  00000000   =  ................
   000080   ===>  00000000  00000000  F1F24BF5  F44BF2F6   =  ........12.54.26
   000090   ===>  E2C5D740  F2F86B40  F1F9F9F4  00000000   =  SEP 28, 1994....
   0000A0   ===>  C5D8E4C9  D3C1E3C5  D9C1D340  E3D9C9C1   =  EQUILATERAL TRIA
   0000B0   ===>  D5C7D3C5  E2000000  00C9E2D6  E2C3C5D3   =  NGLES....ISOSCEL
   0000C0   ===>  C5E240E3  D9C9C1D5  C7D3C5E2  40400000   =  ES TRIANGLES  ..
   0000D0   ===>  0000E2C3  C1D3C5D5  C540E3D9  C9C1D5C7   =  ..SCALENE TRIANG
   0000E0   ===>  D3C5E240  40404000  000000C9  D5E5C1D3   =  LES    ....INVAL
   0000F0   ===>  C9C440E3  D9C9C1D5  C7D3C5E2  40404040   =  ID TRIANGLES
   000100   ===>  00000000  00000000  00000000  00000000   =  ................
   000110   ===>  00000000  00000000  00000000  00000000   =  ................
```

Figure 5-33. Result of Entering the MEMORY Command

The MEMORY command can be entered with indirect register addressing to specify a location. For example, the following command displays memory starting from the location pointed to by the 24-bit mode address in register 13, as seen in Figure 5-34:

```
MEMORY R13%
```

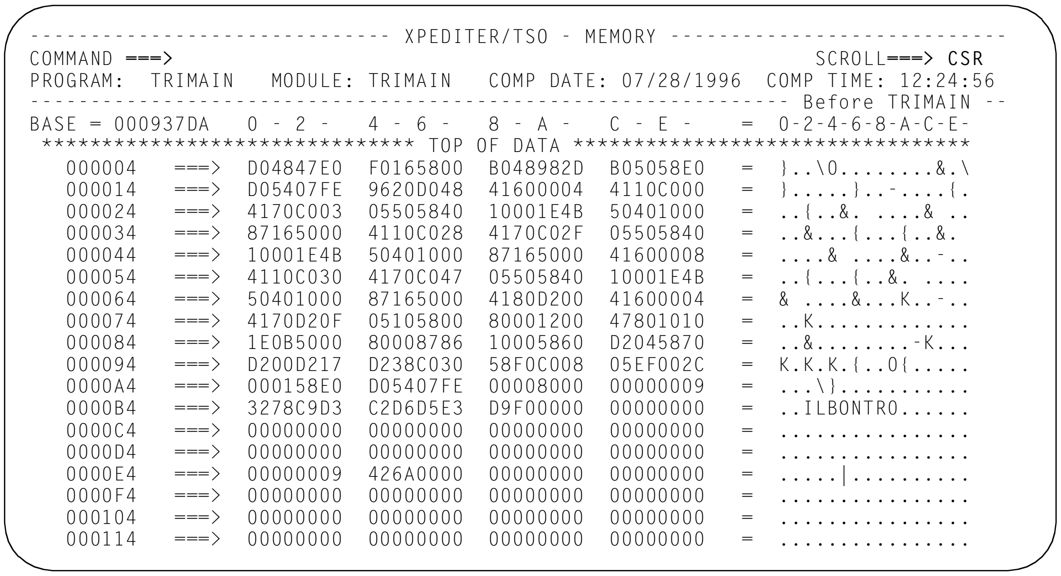


```
------------------------------ XPEDITER/TSO - MEMORY ---------------------------
COMMAND ===>                                                     SCROLL===> CSR
PROGRAM:  TRIMAIN   MODULE: TRIMAIN   COMP DATE: 07/28/1996  COMP TIME: 12:54:26
---------------------------------------------------------------- Before TRIMAIN --
BASE = 00093278   0 - 2 -   4 - 6 -   8 - A -   C - E -    =  0-2-4-6-8-A-C-E-
 ******************************** TOP OF DATA ********************************
   000000   ===>  00300000  00096B68  00000000  00000000   =  .....,..........
   000010   ===>  00000000  00000000  00000000  00000000   =  ................
   000020   ===>  00000000  00000000  00000000  00000000   =  ................
   000030   ===>  00000000  00000000  00000000  00000000   =  ................
   000040   ===>  00000000  00000000  3102A04B  00000000   =  ................
   000050   ===>  00000000  00093544  00000000  00000000   =  ................
   000060   ===>  00000000  00093880  00093A6E  00000000   =  ...........>....
   000070   ===>  50093854  8008BBFC  0005C5F8  00093786   =  &.........E8....
   000080   ===>  50093838  000930D8  00000000  00093488   =  &......Q........
   000090   ===>  000937DA  00093038  00093038  000934E0   =  ...............\
   0000A0   ===>  00000000  00000000  00000000  00000000   =  ................
   0000B0   ===>  00000000  00000000  00000000  00000000   =  ................
   0000C0   ===>  00000000  00000000  00000000  00000000   =  ................
   0000D0   ===>  00000000  00000000  00000000  00000000   =  ................
   0000E0   ===>  00000000  00000000  00000000  00000000   =  ................
   0000F0   ===>  00000000  00000000  00000000  00000000   =  ................
   000100   ===>  00000000  00000000  00000000  00000000   =  ................
   000110   ===>  00000000  00000000  00000000  00000000   =  ................
```

Figure 5-34. Result of Entering MEMORY R13%

Also, you can use arithmetic expressions such as the following to access storage, as shown in Figure 5-35.

```
MEMORY R9%+4
```

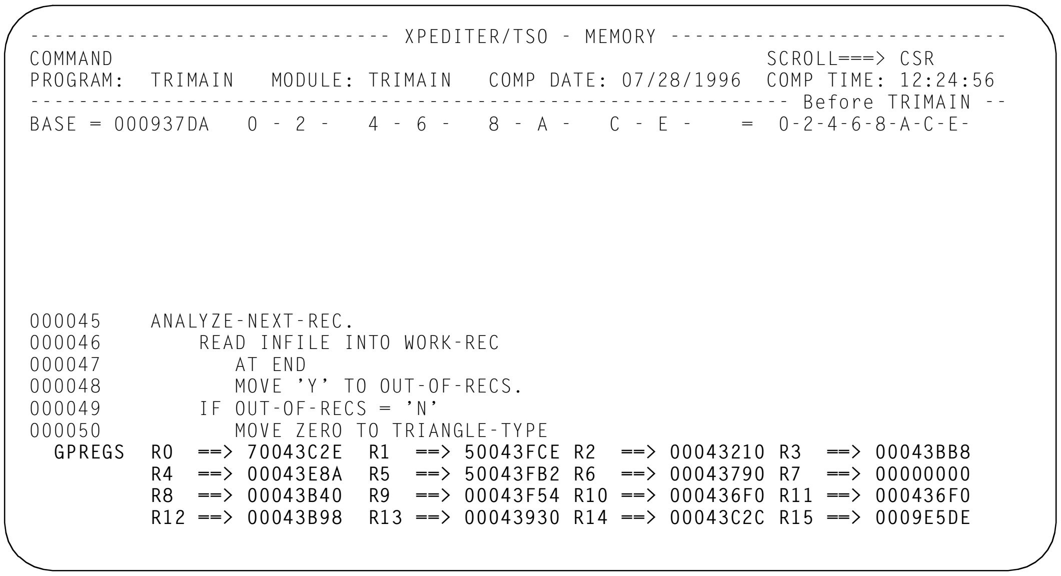


```
------------------------------ XPEDITER/TSO - MEMORY ---------------------------
COMMAND ===>                                                     SCROLL===> CSR
PROGRAM:  TRIMAIN   MODULE: TRIMAIN   COMP DATE: 07/28/1996  COMP TIME: 12:24:56
---------------------------------------------------------------- Before TRIMAIN --
BASE = 000937DA   0 - 2 -   4 - 6 -   8 - A -   C - E -    =  0-2-4-6-8-A-C-E-
 ******************************** TOP OF DATA ********************************
   000004   ===>  D04847E0  F0165800  B048982D  B05058E0   =  }..\0........&.\
   000014   ===>  D05407FE  9620D048  41600004  4110C000   =  }.....}..-....{.
   000024   ===>  4170C003  05505840  10001E4B  50401000   =  ..{..&. ....& ..
   000034   ===>  87165000  4110C028  4170C02F  05505840   =  ..&...{...{..&.
   000044   ===>  10001E4B  50401000  87165000  41600008   =  ....& ....&..-..
   000054   ===>  4110C030  4170C047  05505840  10001E4B   =  ..{...{..&. ....
   000064   ===>  50401000  87165000  4180D200  41600004   =  & ....&...K..-..
   000074   ===>  4170D20F  05105800  80001200  47801010   =  ..K.............
   000084   ===>  1E0B5000  80008786  10005860  D2045870   =  ..&.........-K...
   000094   ===>  D200D217  D238C030  58F0C008  05EF002C   =  K.K.K.{..0{.....
   0000A4   ===>  000158E0  D05407FE  00008000  00000009   =  ...\}...........
   0000B4   ===>  3278C9D3  C2D6D5E3  D9F00000  00000000   =  ..ILBONTR0......
   0000C4   ===>  00000000  00000000  00000000  00000000   =  ................
   0000D4   ===>  00000000  00000000  00000000  00000000   =  ................
   0000E4   ===>  00000009  426A0000  00000000  00000000   =  .....|..........
   0000F4   ===>  00000000  00000000  00000000  00000000   =  ................
   000104   ===>  00000000  00000000  00000000  00000000   =  ................
   000114   ===>  00000000  00000000  00000000  00000000   =  ................
```

Figure 5-35. Result of Entering MEMORY R9%+4

Using the GPREGS Command

The GPREGS command opens a window and displays the 16 general-purpose registers at the bottom of the screen. Figure 5-36 on page 5-21 shows the result of entering the GPREGS command for the TRIMAIN program. The displayed hexadecimal values can be typed over to change the register contents.



```
------------------------------ XPEDITER/TSO - MEMORY ----------------------------
COMMAND                                                      SCROLL===> CSR
PROGRAM:  TRIMAIN   MODULE: TRIMAIN   COMP DATE: 07/28/1996  COMP TIME: 12:24:56
------------------------------------------------------------- Before TRIMAIN --
BASE = 000937DA   0 - 2 -   4 - 6 -   8 - A -   C - E -    =  0-2-4-6-8-A-C-E-









000045    ANALYZE-NEXT-REC.
000046        READ INFILE INTO WORK-REC
000047           AT END
000048           MOVE 'Y' TO OUT-OF-RECS.
000049        IF OUT-OF-RECS = 'N'
000050           MOVE ZERO TO TRIANGLE-TYPE
  GPREGS  R0  ==> 70043C2E  R1  ==> 50043FCE R2  ==> 00043210 R3  ==> 00043BB8
          R4  ==> 00043E8A  R5  ==> 50043FB2 R6  ==> 00043790 R7  ==> 00000000
          R8  ==> 00043B40  R9  ==> 00043F54 R10 ==> 000436F0 R11 ==> 000436F0
          R12 ==> 00043B98  R13 ==> 00043930 R14 ==> 00043C2C R15 ==> 0009E5DE
```

Figure 5-36. Result of Entering the GPREGS Command

The register window can be removed from display by entering the following command:

```
GPREGS OFF
```

Using the TOGGLE Command

The TOGGLE command lets you switch from one panel display to another. For COBOL, you may move back and forth between the Listing screen and the Storage screen. It may be beneficial for you to define a specific PF KEY to efficiently utilize the toggle functionality.

Analyzing Program Logic

This section describes the built-in dynamic analysis features that let you identify program structure, trace the flow of control, monitor execution coverage, and review the execution path. These features assist you in understanding what the program does and how to reach a certain location in the program. The XPEDITER/TSO commands covered here are FIND, TRACE, SHOW PREVIOUS, COUNT, MONITOR, REVERSE, and RESUME.

If you have XPEDITER for DB2 Extension and File-AID for DB2, you can analyze how SQL statements execute with the EXPLAIN command. Refer to “Using XPEDITER for DB2 Extension” on page 5-40 for more information.

Identifying Program Structure

The XPEDITER/TSO FIND command allows you to search data relationships and program structures, in addition to locating character strings. For instance, the FIND command can process data names and identify COBOL statements that directly or indirectly affect or refer to the data names. COBOL-structure keywords such as ALTER, CONDITION, I/O, etc. are processed to query COBOL statements that have the potential to modify data, conditional constructs, and I/O statements. With the highlighting effect and the capability to suppress statements that do not qualify for the search category, the source display screen can turn into a representation of “data flow cross reference” and a “high-level structure.”

One example of the COBOL sensitivity of FIND is the ability to find data names, aliases and the use of the data name. Some of the keywords related to finding data names are:

DEFine	Data name is defined.
MODify	Value of the data name has changed or has the potential for change.
USE	Value of the data name is used, but not modified.
REFerence	Data name is defined, modified, or used.

The default is REFERENCE. When FIND is issued on a data name with no additional keywords, all references to the data name are found.

Finding All References for a Data Name

Enter the following command to find the data name SUBS:

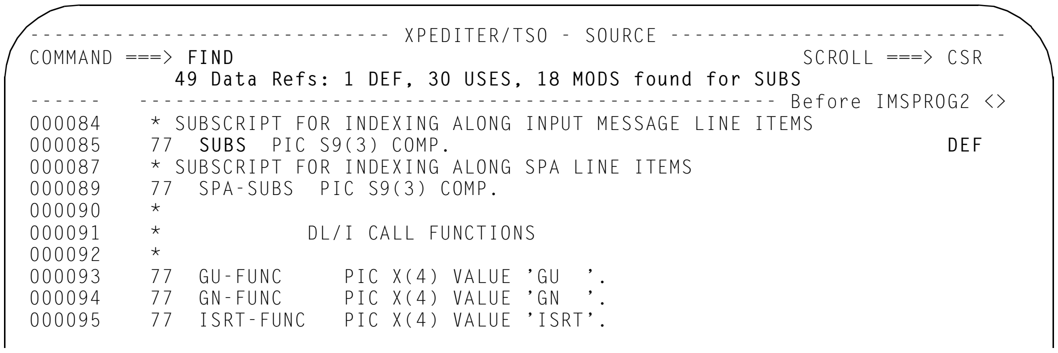
```
FIND SUBS
```

Note: The SET KEEP MAX 5 command was used to suppress the Keep window when no keeps are explicitly requested, and the SET AUTOKEEP OFF command was used to turn off the display of automatically kept data for all the FIND examples shown in this section.

The response to the FIND SUBS command is illustrated in Figure 5-37. The message line indicates the number of times the data name SUBS is referenced in the program. There are 49 data references for SUBS: It is defined once, used 30 times, and modified 18 times. Each reference for the data name SUBS is highlighted and one of the following messages appear in the message area to the right of the found line: DEF, USE, or MOD.

Note: Defines will be found first.All defines are found in the Data Division because this is the section where the data names are defined.The uses and modifications of a data name will be found in the Procedure Division.

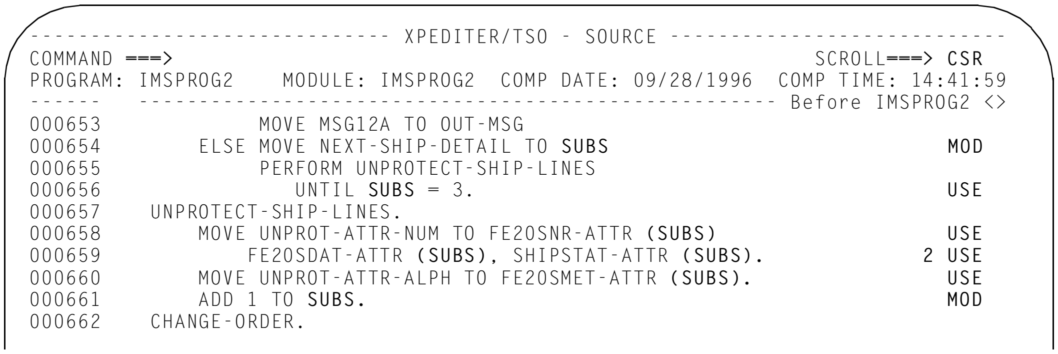
To find the next occurrence of SUBS, press **PF5** or type **FIND** on the command line as shown in Figure 5-37, move the cursor down past the line in which SUBS is defined, and press **Enter**.



```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===> FIND                                              SCROLL ===> CSR
             49 Data Refs: 1 DEF, 30 USES, 18 MODS found for SUBS
------   ---------------------------------------------------- Before IMSPROG2 <>
000084    * SUBSCRIPT FOR INDEXING ALONG INPUT MESSAGE LINE ITEMS
000085    77  SUBS  PIC S9(3) COMP.                                      DEF
000087    * SUBSCRIPT FOR INDEXING ALONG SPA LINE ITEMS
000089    77  SPA-SUBS  PIC S9(3) COMP.
000090    *
000091    *            DL/I CALL FUNCTIONS
000092    *
000093    77  GU-FUNC     PIC X(4) VALUE 'GU  '.
000094    77  GN-FUNC     PIC X(4) VALUE 'GN  '.
000095    77  ISRT-FUNC   PIC X(4) VALUE 'ISRT'.
```

Figure 5-37. Result of Entering FIND SUBS

Figure 5-38 on page 5-23 shows the result of the repeat FIND. The DEF, MOD, and USE messages remain on the display until execution begins or a new FIND command is issued. Note that entry of a repeat FIND does not remove these messages.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                     SCROLL===> CSR
PROGRAM: IMSPROG2    MODULE: IMSPROG2  COMP DATE: 09/28/1996  COMP TIME: 14:41:59
------   ----------------------------------------------------- Before IMSPROG2 <>
000653             MOVE MSG12A TO OUT-MSG
000654        ELSE MOVE NEXT-SHIP-DETAIL TO SUBS                             MOD
000655             PERFORM UNPROTECT-SHIP-LINES
000656                UNTIL SUBS = 3.                                        USE
000657    UNPROTECT-SHIP-LINES.
000658        MOVE UNPROT-ATTR-NUM TO FE20SNR-ATTR (SUBS)                    USE
000659            FE20SDAT-ATTR (SUBS), SHIPSTAT-ATTR (SUBS).              2 USE
000660        MOVE UNPROT-ATTR-ALPH TO FE20SMET-ATTR (SUBS).                 USE
000661        ADD 1 TO SUBS.                                                 MOD
000662    CHANGE-ORDER.
```

Figure 5-38. Result of a Repeat FIND

The remaining keywords that relate to finding data names are the following two groups—ALIAS, NOREDEFINE, NOALIAS, and DIRECT and INDIRECT. They are defined below and illustrated in examples on the following pages.

ALIas	Other references (such as redefined or group level names) to the same storage location are found.
NORedefine	Other references (except redefines) to the same storage location are found.
NOAlias	Alias names are not found; default.
DIRect	Only direct references to the data name are found; default.
INDirect	All references to the data name, its aliases (if indicated), and all places a data value is passed to or from the data name and its aliases are found.

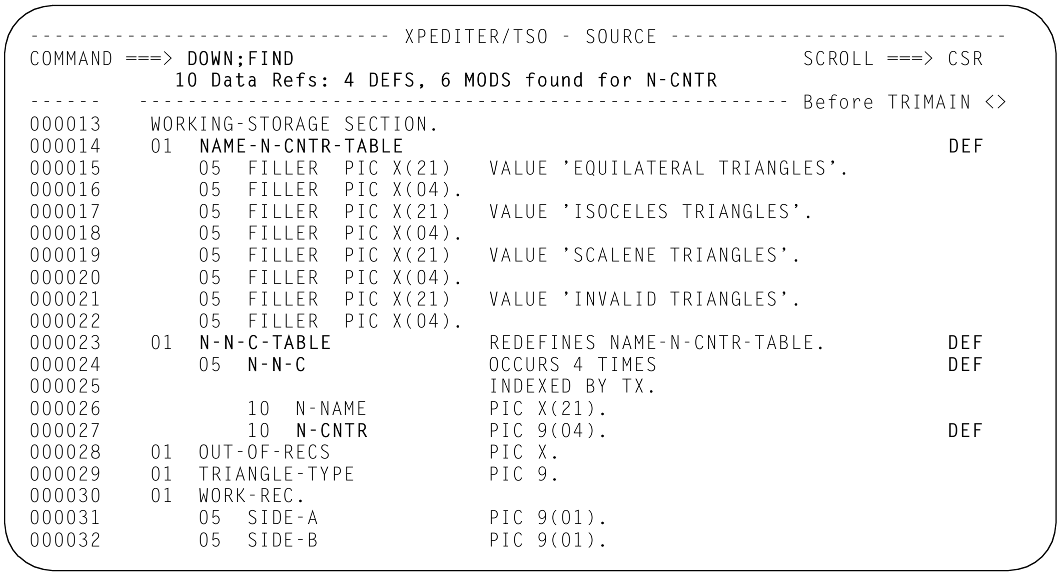
Finding Aliases of a Data Name

The data name N-CNTR has the aliases N-N-C, N-N-C-TABLE, and NAME-N-CNTR-TABLE. Enter the following command to find all aliases of the data name N-CNTR:

```
FIND N-CNTR ALIAS
```

The result of this FIND command is shown inFigure 5-39 on page 5-24, where N-N-C-TABLE and N-N-C are aliases of N-CNTR because they are both group data names under which N-CNTR is defined. NAME-N-CNTR-TABLE is an alias of N-CNTR because N-N-C-TABLE redefines NAME-N-CNTR-TABLE. That is, N-N-C, N-N-C-TABLE, and NAME-N-CNTR-TABLE all have the same storage location as N-CNTR, as seen in the Data Division.

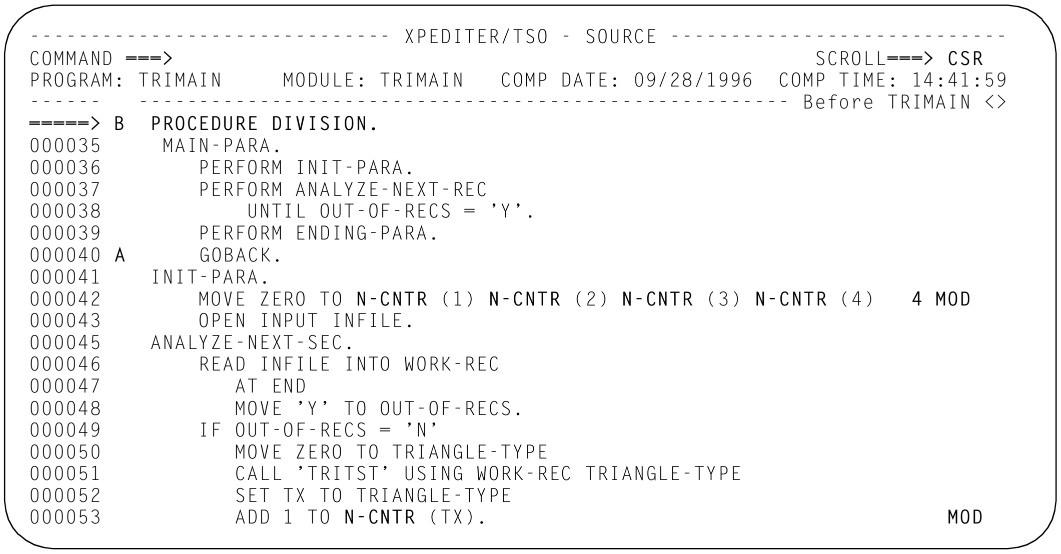
Enter the command **DOWN;FIND**, so the screen will be scrolled before a repeat FIND is issued.



```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===> DOWN;FIND                                         SCROLL ===> CSR
            10 Data Refs: 4 DEFS, 6 MODS found for N-CNTR
------   ---------------------------------------------------- Before TRIMAIN <>
000013    WORKING-STORAGE SECTION.
000014    01  NAME-N-CNTR-TABLE                                            DEF
000015        05  FILLER  PIC X(21)   VALUE 'EQUILATERAL TRIANGLES'.
000016        05  FILLER  PIC X(04).
000017        05  FILLER  PIC X(21)   VALUE 'ISOCELES TRIANGLES'.
000018        05  FILLER  PIC X(04).
000019        05  FILLER  PIC X(21)   VALUE 'SCALENE TRIANGLES'.
000020        05  FILLER  PIC X(04).
000021        05  FILLER  PIC X(21)   VALUE 'INVALID TRIANGLES'.
000022        05  FILLER  PIC X(04).
000023    01  N-N-C-TABLE             REDEFINES NAME-N-CNTR-TABLE.         DEF
000024        05  N-N-C               OCCURS 4 TIMES                       DEF
000025                                INDEXED BY TX.
000026            10  N-NAME          PIC X(21).
000027            10  N-CNTR          PIC 9(04).                           DEF
000028    01  OUT-OF-RECS             PIC X.
000029    01  TRIANGLE-TYPE           PIC 9.
000030    01  WORK-REC.
000031        05  SIDE-A              PIC 9(01).
000032        05  SIDE-B              PIC 9(01).
```

Figure 5-39. Result of Finding N-CNTR With DIRECT and ALIAS

The display scrolls to show the uses and modifications of N-CNTR in the Procedure Division. In Figure 5-40, notice that the number of times N-CNTR is used and modified on the line is indicated on statement 42.



```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                    SCROLL===> CSR
PROGRAM: TRIMAIN     MODULE: TRIMAIN   COMP DATE: 09/28/1996  COMP TIME: 14:41:59
------   ---------------------------------------------------- Before TRIMAIN <>
=====> B  PROCEDURE DIVISION.
000035     MAIN-PARA.
000036        PERFORM INIT-PARA.
000037        PERFORM ANALYZE-NEXT-REC
000038            UNTIL OUT-OF-RECS = 'Y'.
000039        PERFORM ENDING-PARA.
000040 A      GOBACK.
000041    INIT-PARA.
000042        MOVE ZERO TO N-CNTR (1) N-CNTR (2) N-CNTR (3) N-CNTR (4)   4 MOD
000043        OPEN INPUT INFILE.
000045    ANALYZE-NEXT-SEC.
000046        READ INFILE INTO WORK-REC
000047           AT END
000048           MOVE 'Y' TO OUT-OF-RECS.
000049        IF OUT-OF-RECS = 'N'
000050           MOVE ZERO TO TRIANGLE-TYPE
000051           CALL 'TRITST' USING WORK-REC TRIANGLE-TYPE
000052           SET TX TO TRIANGLE-TYPE
000053           ADD 1 TO N-CNTR (TX).                                    MOD
```

Figure 5-40. Scrolling to Modifications of N-CNTR

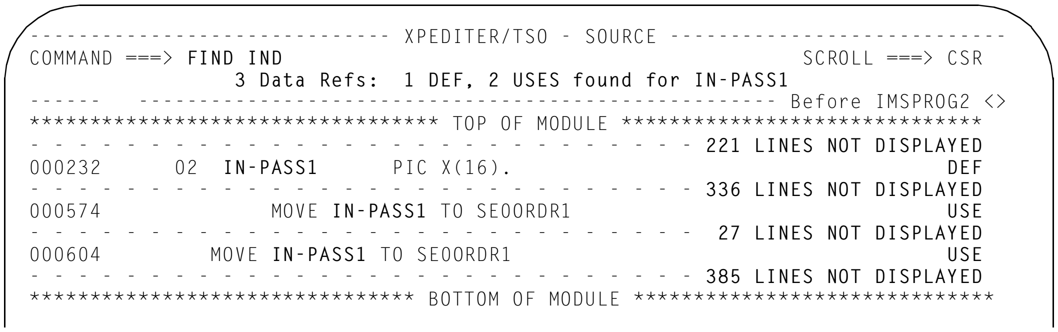
Finding Indirect References to a Data Name

In all of the examples that have been discussed so far, the FIND default of DIRECT was used. When INDIRECT is specified for a data name, all statements directly or indirectly affected by the data name are found. The following example illustrates how INDIRECT is used and the results.

A good way to view indirect references is to use the EXCLUDE keyword with the FIND command.The EXCLUDE keyword excludes from view all lines that were not found. For example, issue the following command for the data name IN-PASS1:

```
FIND IN-PASS1 IND X
```

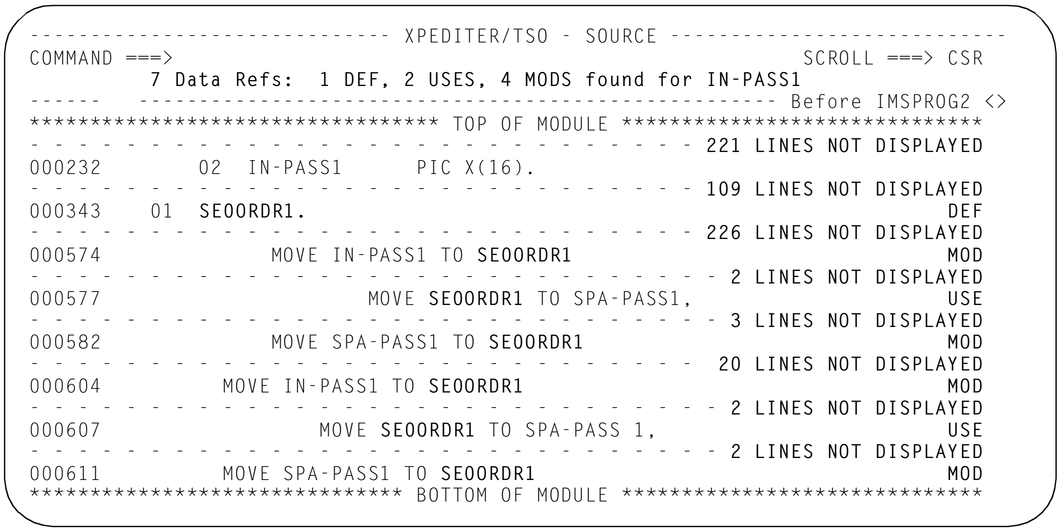
The results of this FIND command are shown inFigure 5-41 where all references to IN-PASS1 are displayed. To display the next level of indirect references, enter **FIND INDIRECT** or press the **PF17** key.



```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===> FIND IND                                           SCROLL ===> CSR
                 3 Data Refs:  1 DEF, 2 USES found for IN-PASS1
------   ----------------------------------------------------- Before IMSPROG2 <>
****************************** TOP OF MODULE *****************************
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - 221 LINES NOT DISPLAYED
000232      02  IN-PASS1      PIC X(16).                                     DEF
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - 336 LINES NOT DISPLAYED
000574              MOVE IN-PASS1 TO SE0ORDR1                                USE
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -  27 LINES NOT DISPLAYED
000604         MOVE IN-PASS1 TO SE0ORDR1                                     USE
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - 385 LINES NOT DISPLAYED
******************************* BOTTOM OF MODULE *****************************
```

Figure 5-41. Finding IN-PASS1 INDIRECT With EXCLUDE

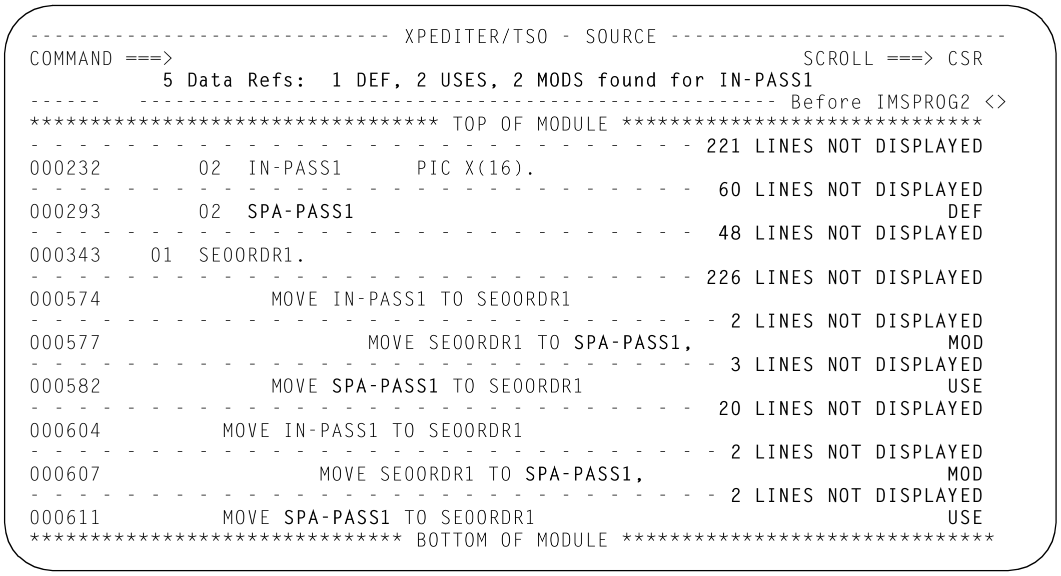
Each time FIND INDIRECT is entered, a new level of indirect references is found. In Figure 5-42, SE0ORDR1 references are highlighted.



```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                    SCROLL ===> CSR
          7 Data Refs:  1 DEF, 2 USES, 4 MODS found for IN-PASS1
------   ----------------------------------------------------- Before IMSPROG2 <>
****************************** TOP OF MODULE *****************************
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - 221 LINES NOT DISPLAYED
000232        02  IN-PASS1      PIC X(16).
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - 109 LINES NOT DISPLAYED
000343    01  SE0ORDR1.                                                      DEF
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - 226 LINES NOT DISPLAYED
000574              MOVE IN-PASS1 TO SE0ORDR1                                MOD
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 2 LINES NOT DISPLAYED
000577                      MOVE SE0ORDR1 TO SPA-PASS1,                      USE
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 3 LINES NOT DISPLAYED
000582              MOVE SPA-PASS1 TO SE0ORDR1                               MOD
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -  20 LINES NOT DISPLAYED
000604         MOVE IN-PASS1 TO SE0ORDR1                                     MOD
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 2 LINES NOT DISPLAYED
000607                  MOVE SE0ORDR1 TO SPA-PASS 1,                         USE
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 2 LINES NOT DISPLAYED
000611         MOVE SPA-PASS1 TO SE0ORDR1                                    MOD
****************************** BOTTOM OF MODULE *****************************
```

Figure 5-42. FIND INDIRECT - First Level of Indirection

When **FIND INDIRECT** (PF17) is entered again, all references to IN-PASS1, SE0ORDR1, and SPA-PASS1 are found (Figure 5-43 on page 5-26).



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                 SCROLL ===> CSR
           5 Data Refs:  1 DEF, 2 USES, 2 MODS found for IN-PASS1
------   ------------------------------------------------------ Before IMSPROG2 <>
********************************* TOP OF MODULE ******************************
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 221 LINES NOT DISPLAYED
000232        02  IN-PASS1      PIC X(16).
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  60 LINES NOT DISPLAYED
000293        02  SPA-PASS1                                                    DEF
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  48 LINES NOT DISPLAYED
000343    01  SEOORDR1.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 226 LINES NOT DISPLAYED
000574              MOVE IN-PASS1 TO SEOORDR1
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 2 LINES NOT DISPLAYED
000577                      MOVE SEOORDR1 TO SPA-PASS1,                        MOD
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 3 LINES NOT DISPLAYED
000582              MOVE SPA-PASS1 TO SEOORDR1                                 USE
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  20 LINES NOT DISPLAYED
000604          MOVE IN-PASS1 TO SEOORDR1
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 2 LINES NOT DISPLAYED
000607                  MOVE SEOORDR1 TO SPA-PASS1,                            MOD
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 2 LINES NOT DISPLAYED
000611          MOVE SPA-PASS1 TO SEOORDR1                                     USE
****************************** BOTTOM OF MODULE ******************************
```

Figure 5-43. FIND INDIRECT - Second Level of Indirection

When no more levels of indirect references are found, the following message is displayed in the message line:

```
END OF INDIRECT SEARCH
```

Enter **END** (PF3) to reset all excluded lines in your program.

Note: The EXCLUDE keyword can be used on any FIND command. When it is used, XPEDITER/TSO excludes all lines before executing the FIND.

Finding COBOL Structures

FIND is also able to find COBOL structures using keywords in place of a character string. When a COBOL-structure keyword is entered instead of a data name, XPEDITER/TSO highlights all lines in the program where the COBOL structure is used.

Note: Currently, EXEC SQL WHENEVER and EXEC SQL DECLARE statements are not found when you issue the FIND SQL command.

The COBOL-structure keywords are listed below. See Appendix A in the *XPEDITER/TSO and XPEDITER/IMS Reference Manual* for descriptions and source relationships.

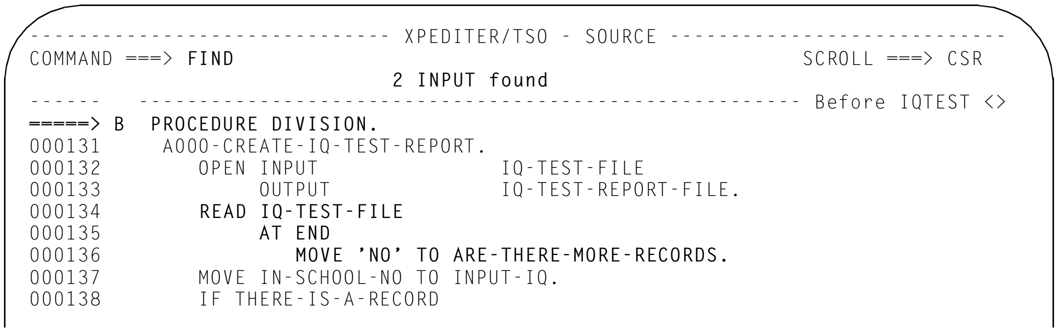
```
ALTer         CICS           INput         PARAgraph
BRAnch        CONDition      IO            SQL
CALL          DLI            OUTput
```

Using FIND With the COBOL-Structure INPUT

Enter the following command to find all INPUT statements in the IQTEST program:

```
FIND INPUT
```

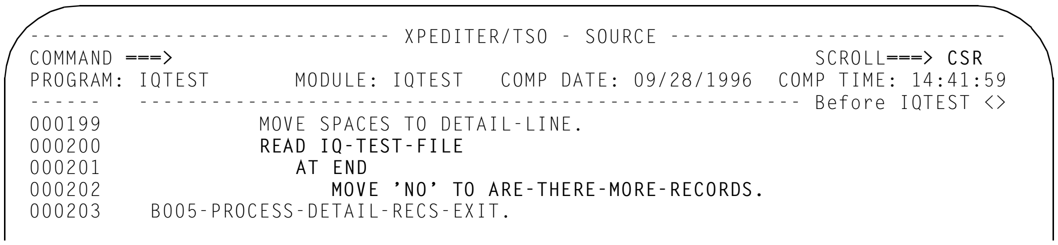
The first input statement found is a READ statement, as shown in Figure 5-44 on page 5-27. The line that contains the input verb is highlighted. If the verb statement extends over multiple lines, multiple lines are highlighted, as shown. To locate the next input statement, press **PF5** (repeat FIND).



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===> FIND                                               SCROLL ===> CSR
                              2 INPUT found
------   ------------------------------------------------------ Before IQTEST <>
=====> B  PROCEDURE DIVISION.
000131     A000-CREATE-IQ-TEST-REPORT.
000132        OPEN INPUT               IQ-TEST-FILE
000133             OUTPUT              IQ-TEST-REPORT-FILE.
000134        READ IQ-TEST-FILE
000135             AT END
000136                MOVE 'NO' TO ARE-THERE-MORE-RECORDS.
000137        MOVE IN-SCHOOL-NO TO INPUT-IQ.
000138        IF THERE-IS-A-RECORD
```

Figure 5-44. Result of FIND With COBOL-Structure Keyword INPUT

The next input statement is also a READ statement as shown in Figure 5-45.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                     SCROLL===> CSR
PROGRAM: IQTEST       MODULE: IQTEST   COMP DATE: 09/28/1996  COMP TIME: 14:41:59
------   ------------------------------------------------------ Before IQTEST <>
000199             MOVE SPACES TO DETAIL-LINE.
000200             READ IQ-TEST-FILE
000201                AT END
000202                   MOVE 'NO' TO ARE-THERE-MORE-RECORDS.
000203    B005-PROCESS-DETAIL-RECS-EXIT.
```

Figure 5-45. Result of a Repeat FIND for INPUT Keyword

When you enter FIND and no more INPUT statements are found, the following message is displayed in the message area:

```
BOTTOM OF DATA REACHED
```

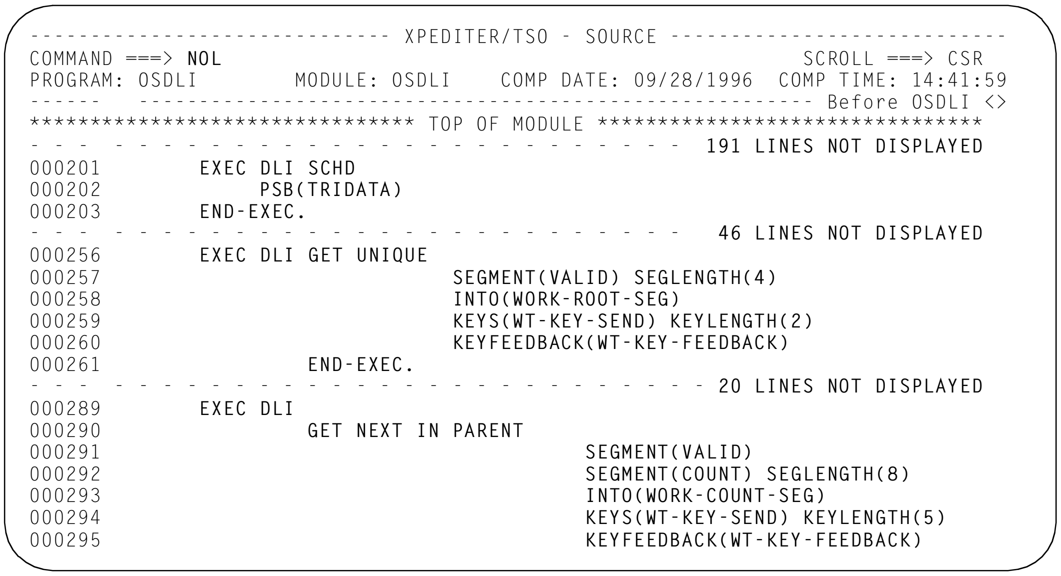
The input statements remain highlighted until execution begins or a new FIND command is issued.

Using FIND With the COBOL-Structure DLI and EXCLUDE Keywords

In this example, the FIND command is entered with the DLI and EXCLUDE keywords. The DLI keyword finds not only all lines in the OSDLI program that say EXEC DLI, but in each case, the entire DLI statement. The EXCLUDE keyword displays only the lines containing the requested information; all other lines are excluded from display. Enter the following command:

```
FIND DLI X
```

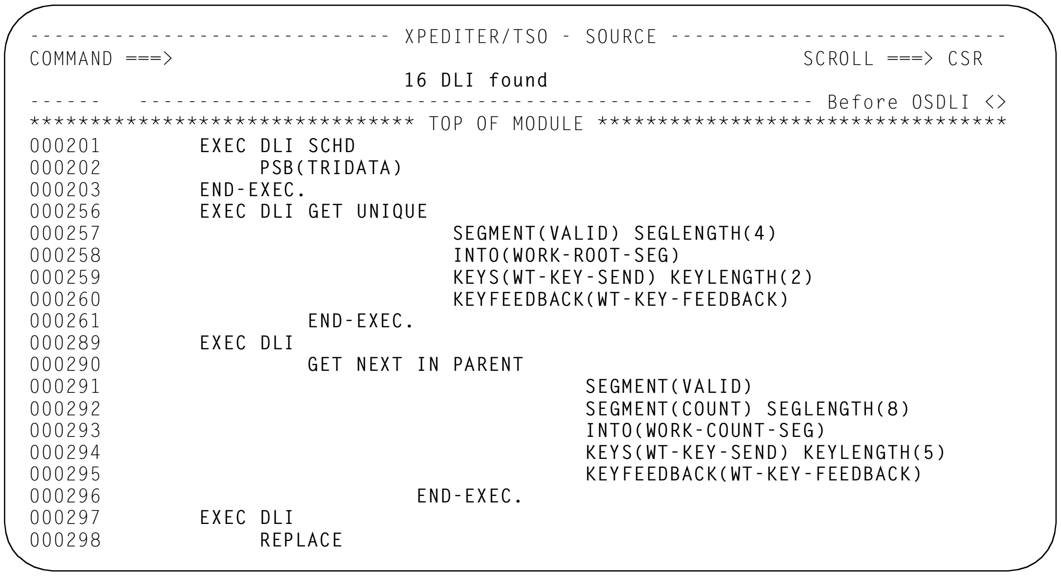
InFigure 5-46 on page 5-28, you can see at a glance where the DLI statements are, what types they are, and all parameters on each statement.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===> NOL                                               SCROLL ===> CSR
PROGRAM: OSDLI        MODULE: OSDLI    COMP DATE: 09/28/1996  COMP TIME: 14:41:59
------  ------------------------------------------------------ Before OSDLI <>
***************************** TOP OF MODULE ********************************
- - -  - - - - - - - - - - - - - - - - - - - - - - - - 191 LINES NOT DISPLAYED
000201        EXEC DLI SCHD
000202             PSB(TRIDATA)
000203        END-EXEC.
- - -  - - - - - - - - - - - - - - - - - - - - - - - -  46 LINES NOT DISPLAYED
000256        EXEC DLI GET UNIQUE
000257                          SEGMENT(VALID) SEGLENGTH(4)
000258                          INTO(WORK-ROOT-SEG)
000259                          KEYS(WT-KEY-SEND) KEYLENGTH(2)
000260                          KEYFEEDBACK(WT-KEY-FEEDBACK)
000261                 END-EXEC.
- - -  - - - - - - - - - - - - - - - - - - - - - - - - - 20 LINES NOT DISPLAYED
000289        EXEC DLI
000290                 GET NEXT IN PARENT
000291                                     SEGMENT(VALID)
000292                                     SEGMENT(COUNT) SEGLENGTH(8)
000293                                     INTO(WORK-COUNT-SEG)
000294                                     KEYS(WT-KEY-SEND) KEYLENGTH(5)
000295                                     KEYFEEDBACK(WT-KEY-FEEDBACK)
```

Figure 5-46. Result of FIND DLI With EXCLUDE

Now, enter a NOLINES command to eliminate the message line `xx LINES NOT DISPLAYED` from the display. As illustrated in Figure 5-47, the resulting display can now accommodate additional found lines.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                   SCROLL ===> CSR
                              16 DLI found
------  ------------------------------------------------------ Before OSDLI <>
******************************* TOP OF MODULE ********************************
000201        EXEC DLI SCHD
000202             PSB(TRIDATA)
000203        END-EXEC.
000256        EXEC DLI GET UNIQUE
000257                          SEGMENT(VALID) SEGLENGTH(4)
000258                          INTO(WORK-ROOT-SEG)
000259                          KEYS(WT-KEY-SEND) KEYLENGTH(2)
000260                          KEYFEEDBACK(WT-KEY-FEEDBACK)
000261                 END-EXEC.
000289        EXEC DLI
000290                 GET NEXT IN PARENT
000291                                     SEGMENT(VALID)
000292                                     SEGMENT(COUNT) SEGLENGTH(8)
000293                                     INTO(WORK-COUNT-SEG)
000294                                     KEYS(WT-KEY-SEND) KEYLENGTH(5)
000295                                     KEYFEEDBACK(WT-KEY-FEEDBACK)
000296                          END-EXEC.
000297        EXEC DLI
000298             REPLACE
```

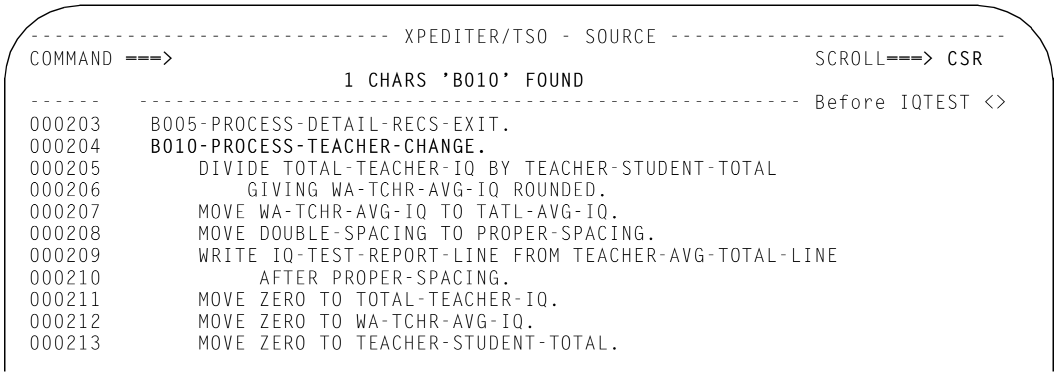
Figure 5-47. Result of Issuing the NOLINES Command

Finding a String IN COBOL Structures

COBOL-structure keywords can also be used with the IN keyword. They are used to find a string or a data name IN a COBOL structure. For example, the following FIND command is entered with the string B010, part of a performed paragraph label:

```
FIND B010 IN PARA
```

The FIND string IN COBOL-structure lets you focus on the statement that is of concern, rather than issue several repeat FIND commands. See the results illustrated in Figure 5-48 on page 5-29.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                     SCROLL===> CSR
                         1 CHARS 'B010' FOUND
------   ----------------------------------------------------- Before IQTEST <>
000203    B005-PROCESS-DETAIL-RECS-EXIT.
000204    B010-PROCESS-TEACHER-CHANGE.
000205        DIVIDE TOTAL-TEACHER-IQ BY TEACHER-STUDENT-TOTAL
000206            GIVING WA-TCHR-AVG-IQ ROUNDED.
000207        MOVE WA-TCHR-AVG-IQ TO TATL-AVG-IQ.
000208        MOVE DOUBLE-SPACING TO PROPER-SPACING.
000209        WRITE IQ-TEST-REPORT-LINE FROM TEACHER-AVG-TOTAL-LINE
000210             AFTER PROPER-SPACING.
000211        MOVE ZERO TO TOTAL-TEACHER-IQ.
000212        MOVE ZERO TO WA-TCHR-AVG-IQ.
000213        MOVE ZERO TO TEACHER-STUDENT-TOTAL.
```

Figure 5-48. Result of FIND Data String With IN PARAGRAPH

The NOLINES Keyword and Command

In the previous example, the NOLINES command was entered to suppress the `xxx LINES NOT DISPLAYED` message line that appears when the EXCLUDE keyword is used with FIND. A NOLINES *keyword* is also available with FIND. Like the NOLINES command, the NOLINES keyword eliminates the message line that appears with the use of the EXCLUDE keyword. However, it is effective only when used in conjunction with the EXCLUDE keyword.

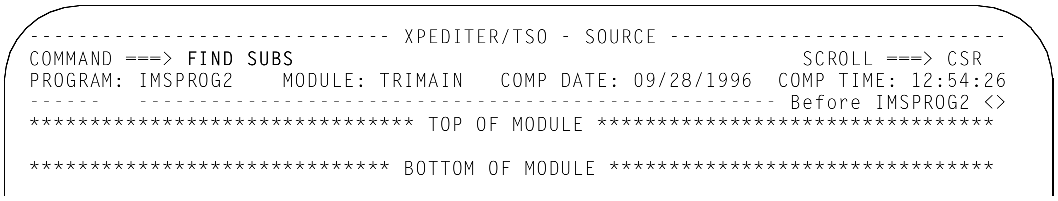
Using the EXCLUDE Command With FIND

The EXCLUDE command includes the keyword parameter ALL, which excludes all lines in a program. The EXCLUDE ALL command can be used effectively with FIND.Issuing the EXCLUDE ALL command results in the removal of all the source lines in the display. You can also use the NOLINES command to suppress the `xxx LINES NOT DISPLAYED` message line.

Entering the following commands results in the display shown in Figure 5-49:

```
EXCLUDE ALL;NOLINES
```

After all lines in the program have been excluded, the FIND command can be issued for multiple data names, to make a cumulative search for the source of a problem.

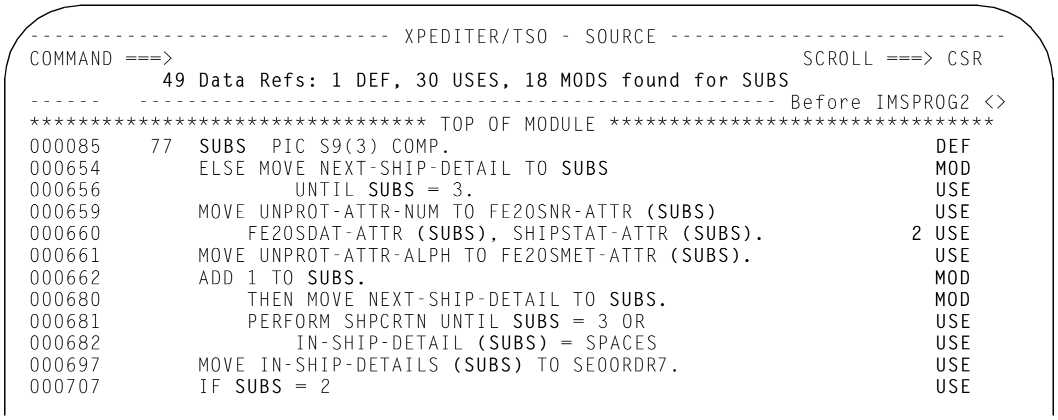


```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===> FIND SUBS                                          SCROLL ===> CSR
PROGRAM: IMSPROG2    MODULE: TRIMAIN   COMP DATE: 09/28/1996  COMP TIME: 12:54:26
------   --------------------------------------------------- Before IMSPROG2 <>
******************************** TOP OF MODULE *********************************

****************************** BOTTOM OF MODULE ********************************
```

Figure 5-49. Result of Excluding ALL Lines

Note that you can also concatenate the EXCLUDE ALL command along with a FIND command. For example, entering the following commands results in the display shown in Figure 5-50 on page 5-30:

```
EXCLUDE ALL;NOLINES;FIND SUBS
```



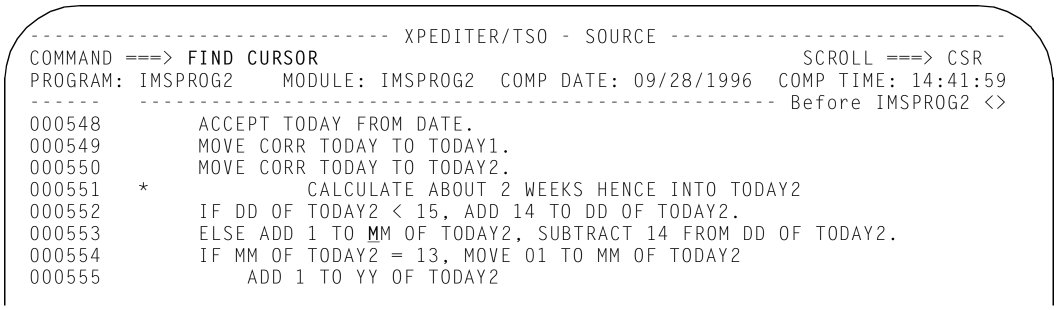
```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                    SCROLL ===> CSR
           49 Data Refs: 1 DEF, 30 USES, 18 MODS found for SUBS
------   ------------------------------------------------------ Before IMSPROG2 <>
******************************** TOP OF MODULE ********************************
000085    77  SUBS  PIC S9(3) COMP.                                          DEF
000654        ELSE MOVE NEXT-SHIP-DETAIL TO SUBS                             MOD
000656                UNTIL SUBS = 3.                                        USE
000659        MOVE UNPROT-ATTR-NUM TO FE2OSNR-ATTR (SUBS)                    USE
000660            FE2OSDAT-ATTR (SUBS), SHIPSTAT-ATTR (SUBS).              2 USE
000661        MOVE UNPROT-ATTR-ALPH TO FE2OSMET-ATTR (SUBS).                 USE
000662        ADD 1 TO SUBS.                                                 MOD
000680            THEN MOVE NEXT-SHIP-DETAIL TO SUBS.                        MOD
000681            PERFORM SHPCRTN UNTIL SUBS = 3 OR                          USE
000682                IN-SHIP-DETAIL (SUBS) = SPACES                         USE
000697        MOVE IN-SHIP-DETAILS (SUBS) TO SEOORDR7.                       USE
000707        IF SUBS = 2                                                    USE
```

Figure 5-50. Result of FIND SUBS Command After EXCLUDE ALL Command

Using the FIND CURSOR

The CURSOR keyword issues a FIND command for the data name or string under the cursor. It functions like PEEK CURSOR, searching for data names as well as strings. IN and OF qualifications are automatically picked up.

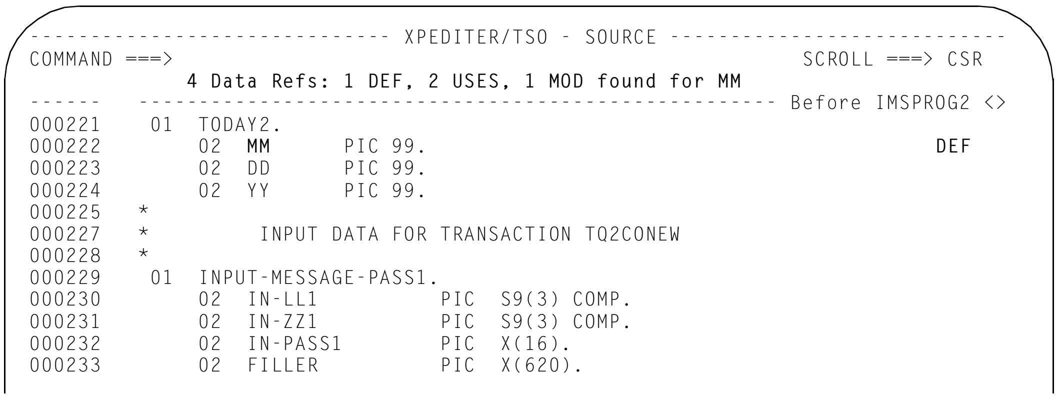
In Figure 5-51, the cursor is positioned on the first M on line 553. Press the **PF14** key or enter the **FIND CURSOR** command as shown.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===> FIND CURSOR                                        SCROLL ===> CSR
PROGRAM: IMSPROG2    MODULE: IMSPROG2  COMP DATE: 09/28/1996  COMP TIME: 14:41:59
------   ------------------------------------------------------ Before IMSPROG2 <>
000548        ACCEPT TODAY FROM DATE.
000549        MOVE CORR TODAY TO TODAY1.
000550        MOVE CORR TODAY TO TODAY2.
000551   *             CALCULATE ABOUT 2 WEEKS HENCE INTO TODAY2
000552        IF DD OF TODAY2 < 15, ADD 14 TO DD OF TODAY2.
000553        ELSE ADD 1 TO MM OF TODAY2, SUBTRACT 14 FROM DD OF TODAY2.
000554        IF MM OF TODAY2 = 13, MOVE 01 TO MM OF TODAY2
000555            ADD 1 TO YY OF TODAY2
```

Figure 5-51. Using the FIND CURSOR

FIND CURSOR searches for the first instance of the data name under the cursor and highlights it, as shown in Figure 5-52.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                    SCROLL ===> CSR
             4 Data Refs: 1 DEF, 2 USES, 1 MOD found for MM
------   ------------------------------------------------------ Before IMSPROG2 <>
000221    01  TODAY2.
000222        02  MM      PIC 99.                                            DEF
000223        02  DD      PIC 99.
000224        02  YY      PIC 99.
000225   *
000227   *         INPUT DATA FOR TRANSACTION TQ2CONEW
000228   *
000229    01  INPUT-MESSAGE-PASS1.
000230        02  IN-LL1          PIC  S9(3) COMP.
000231        02  IN-ZZ1          PIC  S9(3) COMP.
000232        02  IN-PASS1        PIC  X(16).
000233        02  FILLER          PIC  X(620).
```

Figure 5-52. Result of Entering FIND CURSOR

Logging The Results Of A FIND Command

Results of Data Name and COBOL Structure FINDs can optionally be written to the log. Since this option defaults to OFF, a user must enter the SET command as follows to activate this feature:

```
SET LOG FIND ON
```

Information written to the log includes the FIND command entered, the name of the program being searched, the number of occurrences detected, and the source lines containing the argument. For all indirect FINDs and Enhanced FINDs, the level of indirection is also logged.

All source statements containing the argument are displayed after the initial command has been issued, regardless whether the NEXT, PREV, LAST, or FIRST keywords are used. No additional logging will occur when a repeat FIND command is issued. Caution should be used when the FIND command is frequently issued, with explicit attention given to the amount of space allocated to the log file.

The general format of the logged data can be seen in the following example:

```
******************************** TOP OF DATA *********************************
+----------------------------------------------------------------------------+
:                                                              JOB: USER123    :
: XPEDITER/TSO RELEASE 06.40.00      CUSTOMER # 010000        STEP: TSOSTEP1   :
: TAPE CREATE DATE 1997050                                    DATE: 03/08/1997 :
: COMPUWARE CORPORATION                                       TIME: 15.16.19   :
:                                                                              :
+----------------------------------------------------------------------------+
   XPED TSO SPF
   TEST TRIMAIN
*** TRIMAIN  FROM USER!.LOADLIB                                  LINK 01/23/1997
   BEFORE TRIMAIN::TRIMAIN:
   AFTER  TRIMAIN::TRIMAIN:
   PAUSE Before TRIMAIN
   BEFORE BREAKPOINT ENCOUNTERED
   SET LOG FIND ON
   FIND TRIANGLE-TYPE
   PROGRAM=TRIMAIN

4 DATA REFS: 1 DEF, 1 USE, 2 MODS FOUND FOR TRIANGLE-TYPE

 000029    01  TRIANGLE-TYPE          PIC 9.                              DEF
 000050           MOVE ZERO TO TRIANGLE-TYPE                              MOD
 000051           CALL 'TRITST' USING WORK-REC TRIANGLE-TYPE              MOD
 000052           SET TX TO TRIANGLE-TYPE                                 USE
******************************* BOTTOM OF DATA *******************************
```

Figure 5-53. Data Format When Logging The Results Of A FIND Command

Tracing the Flow of Control

You can trace the flow of control with the TRACE and the SHOW PREVIOUS commands.

Using the TRACE Command

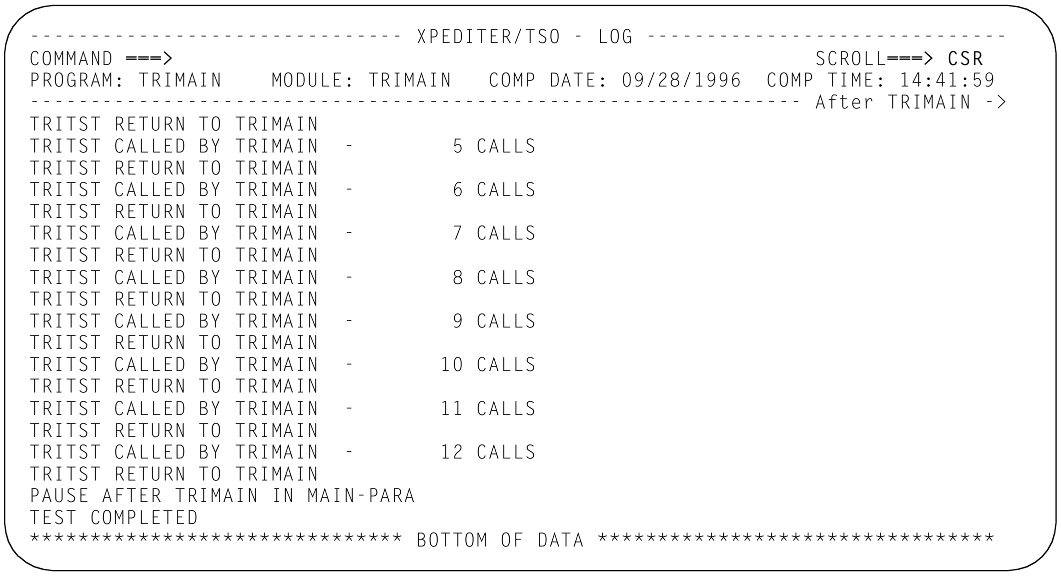
The TRACE command traces the flow of control as your program executes and lets you view it on the Source display screen. The specified statements or paragraph names are highlighted as they are executed, until a breakpoint is reached, an abend is intercepted, a terminal I/O is issued, a keyboard interrupt is detected, or the end of the program is encountered.

In the case of tracing module calls, however, the tracing is not visible on the source screen. The calling module, the called module, and the number of times the calls are made are written to the log during program execution.The call activities can be viewed by entering **LOG** and reviewing the log.

For example, suppose you enter the following TRACE command for the TRIMAIN program:

```
TRACE MODULES
```

After you have executed the program by entering **GO**, you can access the log using the LOG command. As seen in Figure 5-54 on page 5-32, you can trace the flow of control as the various modules are called.



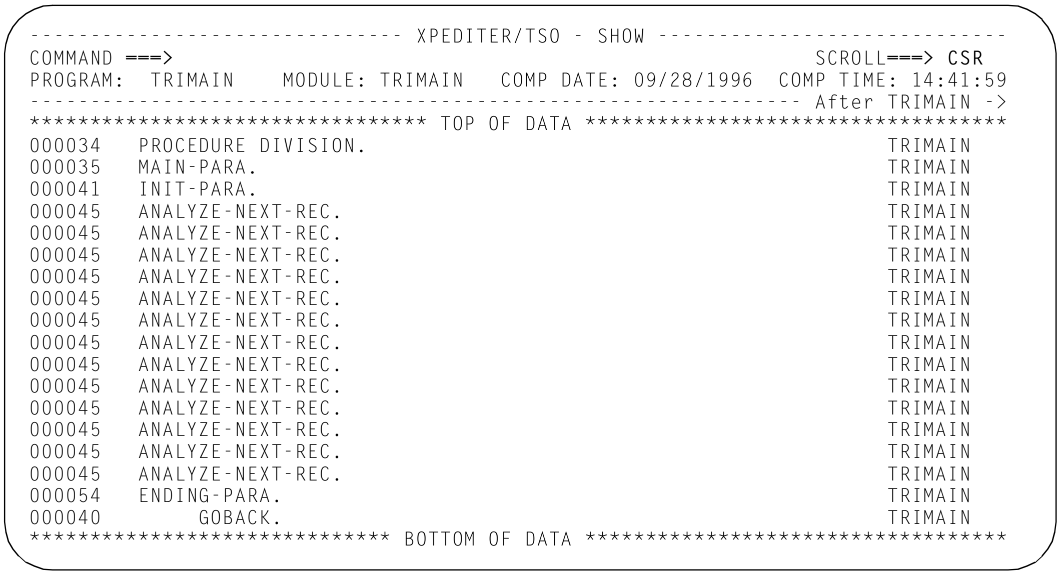
```
------------------------------ XPEDITER/TSO - LOG -----------------------------
COMMAND ===>                                                   SCROLL===> CSR
PROGRAM: TRIMAIN    MODULE: TRIMAIN   COMP DATE: 09/28/1996  COMP TIME: 14:41:59
---------------------------------------------------------------- After TRIMAIN ->
TRITST RETURN TO TRIMAIN
TRITST CALLED BY TRIMAIN  -        5 CALLS
TRITST RETURN TO TRIMAIN
TRITST CALLED BY TRIMAIN  -        6 CALLS
TRITST RETURN TO TRIMAIN
TRITST CALLED BY TRIMAIN  -        7 CALLS
TRITST RETURN TO TRIMAIN
TRITST CALLED BY TRIMAIN  -        8 CALLS
TRITST RETURN TO TRIMAIN
TRITST CALLED BY TRIMAIN  -        9 CALLS
TRITST RETURN TO TRIMAIN
TRITST CALLED BY TRIMAIN  -       10 CALLS
TRITST RETURN TO TRIMAIN
TRITST CALLED BY TRIMAIN  -       11 CALLS
TRITST RETURN TO TRIMAIN
TRITST CALLED BY TRIMAIN  -       12 CALLS
TRITST RETURN TO TRIMAIN
PAUSE AFTER TRIMAIN IN MAIN-PARA
TEST COMPLETED
****************************** BOTTOM OF DATA ********************************
```

Figure 5-54. Session Log for the TRACE MODULES Command

When the TRACE command is used with the MAX keyword, the trace function pauses when execution reaches the preset limit. The default value for the maximum limit is 25. For additional information on the TRACE command refer to “Using the TRACE Command” on page 5-10.

Using the SHOW PREVIOUS Command

The SHOW PREVIOUS command lists, in logical sequence, the previous 100 statements along with the executed breakpoints. The list presented by the SHOW PREVIOUS command can be useful in reviewing the execution path to understand how you got to the present location. Since any implied breakpoints are recognized with the SHOW PREVIOUS command, a program that was run with the TRACE ALL PARAGRAPHS command presents a list like that shown in Figure 5-55 as the result of entering **SHOW PREVIOUS**.



```
------------------------------ XPEDITER/TSO - SHOW ----------------------------
COMMAND ===>                                                   SCROLL===> CSR
PROGRAM:  TRIMAIN    MODULE: TRIMAIN   COMP DATE: 09/28/1996  COMP TIME: 14:41:59
---------------------------------------------------------------- After TRIMAIN ->
******************************** TOP OF DATA **********************************
000034   PROCEDURE DIVISION.                                           TRIMAIN
000035   MAIN-PARA.                                                    TRIMAIN
000041   INIT-PARA.                                                    TRIMAIN
000045   ANALYZE-NEXT-REC.                                             TRIMAIN
000045   ANALYZE-NEXT-REC.                                             TRIMAIN
000045   ANALYZE-NEXT-REC.                                             TRIMAIN
000045   ANALYZE-NEXT-REC.                                             TRIMAIN
000045   ANALYZE-NEXT-REC.                                             TRIMAIN
000045   ANALYZE-NEXT-REC.                                             TRIMAIN
000045   ANALYZE-NEXT-REC.                                             TRIMAIN
000045   ANALYZE-NEXT-REC.                                             TRIMAIN
000045   ANALYZE-NEXT-REC.                                             TRIMAIN
000045   ANALYZE-NEXT-REC.                                             TRIMAIN
000045   ANALYZE-NEXT-REC.                                             TRIMAIN
000045   ANALYZE-NEXT-REC.                                             TRIMAIN
000045   ANALYZE-NEXT-REC.                                             TRIMAIN
000054   ENDING-PARA.                                                  TRIMAIN
000040        GOBACK.                                                  TRIMAIN
****************************** BOTTOM OF DATA ********************************
```

Figure 5-55. Result of Entering the SHOW PREVIOUS Command

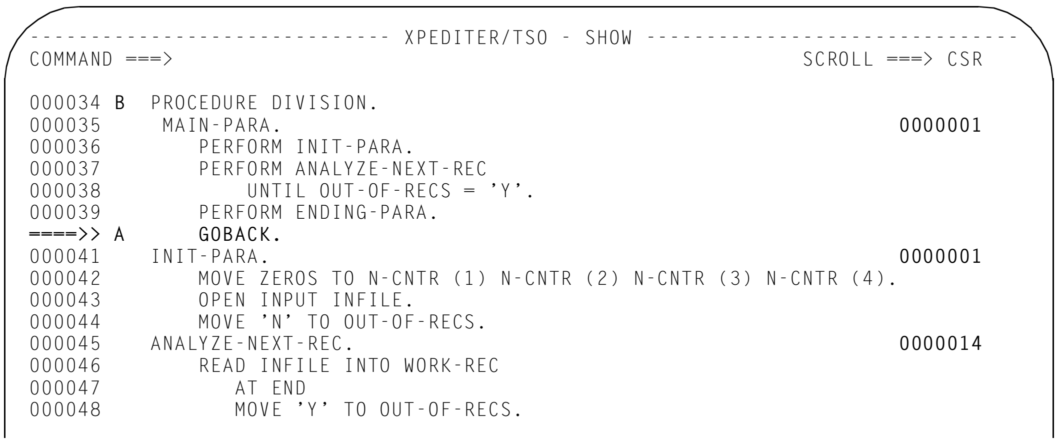
Monitoring Execution Coverage

You can monitor execution coverage with the COUNT and SHOW COUNT commands.

Using the COUNT Command

The COUNT command maintains execution counts and lets you analyze statement level execution coverage after running the program. Figure 5-56 shows the result of setting counters at every paragraph by entering the following command, then pressing **PF12** (GO) to resume execution:

```
COUNT ALL PARAGRAPHS
```



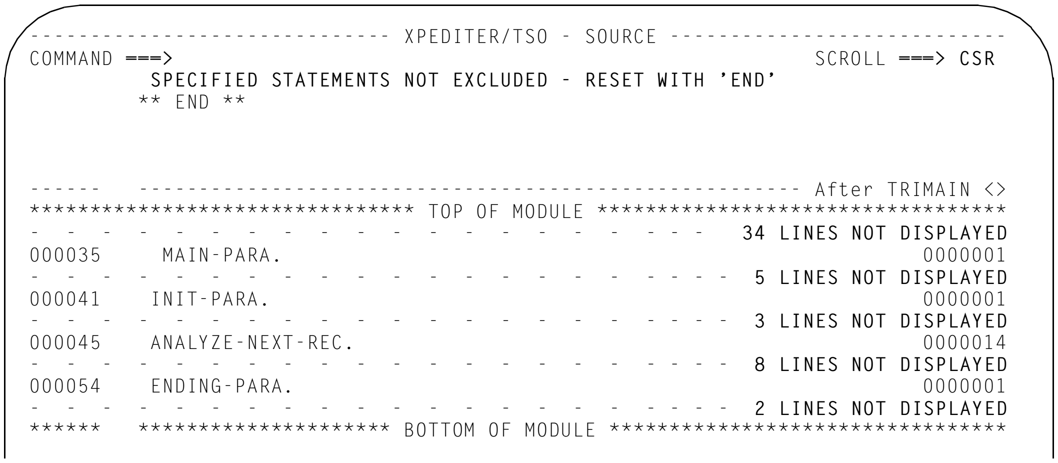
```
------------------------------ XPEDITER/TSO - SHOW ------------------------------
COMMAND ===>                                                      SCROLL ===> CSR

000034 B  PROCEDURE DIVISION.
000035     MAIN-PARA.                                                      0000001
000036        PERFORM INIT-PARA.
000037        PERFORM ANALYZE-NEXT-REC
000038            UNTIL OUT-OF-RECS = 'Y'.
000039        PERFORM ENDING-PARA.
====>> A      GOBACK.
000041    INIT-PARA.                                                       0000001
000042        MOVE ZEROS TO N-CNTR (1) N-CNTR (2) N-CNTR (3) N-CNTR (4).
000043        OPEN INPUT INFILE.
000044        MOVE 'N' TO OUT-OF-RECS.
000045    ANALYZE-NEXT-REC.                                                0000014
000046        READ INFILE INTO WORK-REC
000047           AT END
000048           MOVE 'Y' TO OUT-OF-RECS.
```

Figure 5-56. Result of Entering the COUNT ALL PARAGRAPHS and GO Commands

A 7-digit counter is displayed at the right of the screen for each statement or paragraph that is counted. You can globally monitor execution coverage by using the ALL keyword or selectively monitor by specifying the statement numbers. When the ALL keyword is issued, however, it only applies to the current module.

Using the SHOW COUNT Command

The statements that are monitored with the COUNT command can be listed by entering the SHOW COUNT command. The source lines without counters are excluded from the screen display. Figure 5-57 illustrates the result of the SHOW COUNT command after monitoring execution coverage at the paragraph level.



```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                       SCROLL ===> CSR
          SPECIFIED STATEMENTS NOT EXCLUDED - RESET WITH 'END'
         ** END **



------   ------------------------------------------------------- After TRIMAIN <>
******************************** TOP OF MODULE *********************************
-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - - -  34 LINES NOT DISPLAYED
000035     MAIN-PARA.                                                       0000001
-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - - - -  5 LINES NOT DISPLAYED
000041    INIT-PARA.                                                        0000001
-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - - - -  3 LINES NOT DISPLAYED
000045    ANALYZE-NEXT-REC.                                                 0000014
-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - - - -  8 LINES NOT DISPLAYED
000054    ENDING-PARA.                                                      0000001
-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - - - -  2 LINES NOT DISPLAYED
******   ******************** BOTTOM OF MODULE ********************************
```

Figure 5-57. Result of Entering the SHOW COUNT Command

When the SHOW COUNT command is issued, the entire Procedure Division is written to the log with the 7-digit counters displayed.

If you do not want to write the entire Procedure Division to the log, enter the SHOW COUNT command with the NOLOG keyword. The results of execution coverage will not be recorded.

Monitoring and Reviewing the Execution Path

The MONITOR and REVERSE commands are used to activate review mode. Review mode lets you monitor and review the execution path by stepping backwards through your program. You can view the statements that were executed during normal (forward) execution. You can trace backwards through the *actual sequence of instructions* that led to the current breakpoint and see the data values as they were at the time. There is no guesswork about which of the possible paths the program took; the *actual* path that was taken during forward execution of the program is displayed.

The MONITOR command records the execution history and the REVERSE command enables you to review the execution history.

To activate review mode, enter the MONITOR command from the primary command line. MONITOR without a module name records history for the current module—not necessarily the active module that is currently executing, but the module indicated by the program field on the third line.

After issuing the MONITOR command, execute the statements you want to review. Then, when your program pauses during logical execution, enter the following primary command:

```
REVERSE
```

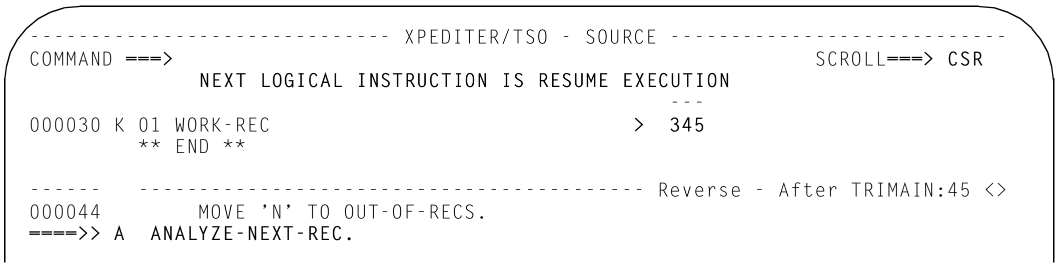
The REVERSE command places the execution arrow on the last statement that was executed, and changes the execution direction of your program—from forward to reverse (backward). From this point on, the REVERSE command acts as a toggle that changes the direction in which your program is executed. During review mode, the execution status message on the fourth line of the screen indicates the execution direction and the statement where execution is paused.

Notes:

1. Entering the REVERSE command only changes the direction of execution; it does not cause execution to occur.
2. You must enter the GO *n* or GO command to begin execution in the current direction.

   The GO *n* command moves the active arrow *n* lines in the current direction, which lets you step through the program line-by-line. Unlike normal execution mode, a GO *n* command in review mode ignores module boundaries and will pause after executing *n* statements, regardless of what modules they are in. It is recommended that you use GO 1 commands to do a backward line-by-line execution.
3. In review mode, TRACE does not recognize the default maximum limit of 25 statements, and continues execution until it encounters the `AT INITIAL EXECUTION POSITION`. To halt the reverse trace, press the **Attn** key.

While review mode is activated, you can set and remove breakpoints, perform tracing (in either direction), and display data.For example, you may want to open a Keep window to view the data values as they are restored to their original state as reverse execution is performed, as shown in Figure 5-58 for the data value WORK-REC.



```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                    SCROLL===> CSR
              NEXT LOGICAL INSTRUCTION IS RESUME EXECUTION
                                                    ---
000030 K 01 WORK-REC                             >  345
         ** END **

------   ----------------------------------------- Reverse - After TRIMAIN:45 <>
000044        MOVE 'N' TO OUT-OF-RECS.
====>> A  ANALYZE-NEXT-REC.
```

Figure 5-58. Review Mode with a Keep Window Opened

Note: While in review mode, you cannot use the GOTO command or alter data by typing over it or by using the MOVE command. Also, skipped lines are ignored during review mode.

To exit from review mode when you have finished doing your analysis, use the RESUME command.

The message `AT CURRENT EXECUTION POSITION` is displayed, and review mode is automatically ended. Normal forward execution occurs until you again enter the REVERSE command.You can also terminate review mode while in forward execution by entering **GO** or **GO** *n* until you see the message `AT CURRENT EXECUTION POSITION`.

Kept items are logged at each breakpoint while in review mode, just as in normal execution mode. The logged items are all logged independently of review mode or normal execution mode, except for the following:

- Encountering the beginning of the program while in review mode.
- Encountering the current execution location while in review mode.

Modifying Program Logic

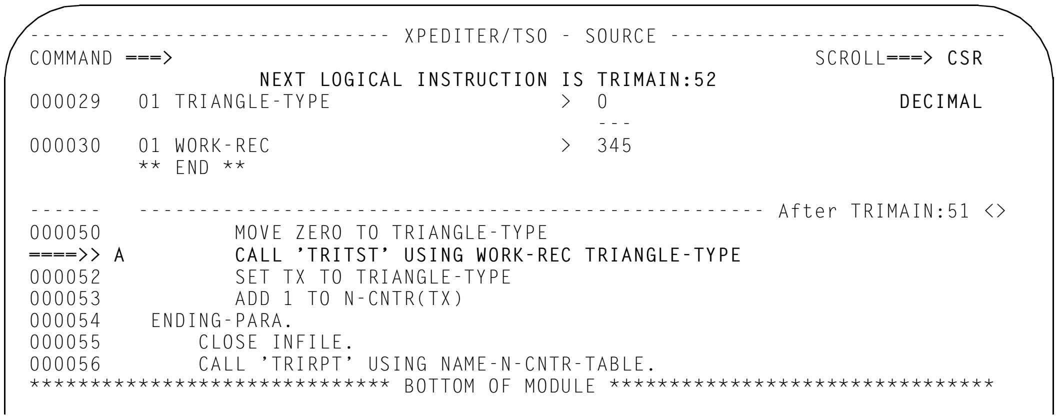
This section describes the XPEDITER/TSO commands that let you modify program logic by bypassing code segments, adding statements, and forcing logic changes.The SKIP, INSERT, GOTO, and MOVE commands let you try out fixes dynamically without requiring any source code modification.

Bypassing Code With the SKIP Command

Unwanted code can be bypassed using the SKIP command. There is no need to comment out the code and recompile the program for it to take effect. For example, a call to a submodule that is not yet written can be bypassed without requiring a program stub to be developed. Figure 5-59 shows the effect of skipping module TRITST and pausing execution following the CALL statement.The following commands were issued:

```
SKIP TRITST:
AFTER 51
GO
```

The effect of the SKIP command can be seen in the parameters WORK-REC and TRIANGLE-TYPE as they are displayed automatically in the Keep window. You could also issue KEEP commands for TRIANGLE-TYPE and WORK-REC to continuously display these parameters.



```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                    SCROLL===> CSR
                NEXT LOGICAL INSTRUCTION IS TRIMAIN:52
000029   01 TRIANGLE-TYPE                 >  0                       DECIMAL
                                             ---
000030   01 WORK-REC                      >  345
         ** END **

------   --------------------------------------------------- After TRIMAIN:51 <>
000050           MOVE ZERO TO TRIANGLE-TYPE
====>> A         CALL 'TRITST' USING WORK-REC TRIANGLE-TYPE
000052           SET TX TO TRIANGLE-TYPE
000053           ADD 1 TO N-CNTR(TX)
000054    ENDING-PARA.
000055        CLOSE INFILE.
000056        CALL 'TRIRPT' USING NAME-N-CNTR-TABLE.
***************************** BOTTOM OF MODULE *******************************
```

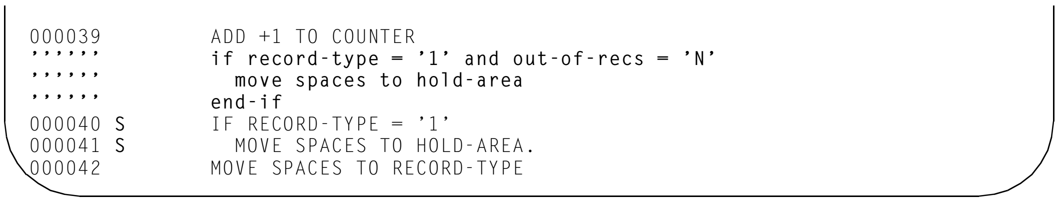
Figure 5-59. Result of Entering the SKIP TRITST: Command

The parameter TRIANGLE-TYPE is left untouched since the CALL to TRITST was bypassed. You can avoid a S0C7 abend by moving a valid value to TRIANGLE-TYPE.

The SKIP command can be used with a statement, a range of statements, a paragraph name, or a module name.

There are a couple of situations to be aware of when using the SKIP command.In order to bypass an entire IF statement, you must skip each verb, not just the statement containing the IF condition.Also, if you skip a statement that sets a switch or flag, the execution path could change or end in an infinite loop. More subtly, the COBOL compiler generates multiple instructions for each COBOL verb. Some of these instructions can load base pointers and base registers for statements. Since a SKIP bypasses all instructions associated with the verb, a S0C1 or S0C4 can result.The SKIP command should be deleted in this case. Use the DELETE SKIP command or the DS line command to delete the skip.

The SKIP command can be combined with inserted statements to test alternative logic flow. Figure 5-60 shows that the original IF statement starting with statement 40 was skipped and completely replaced by the inserted IF logic above it. The insert lines (’’’’’’) were opened up by issuing an I (Insert) line command (I 3) on line 39. Refer to “Inserting Statements” on page 5-36 below for more information on inserting statements.



```
000039          ADD +1 TO COUNTER
''''''          if record-type = '1' and out-of-recs = 'N'
''''''            move spaces to hold-area
''''''          end-if
000040 S        IF RECORD-TYPE = '1'
000041 S          MOVE SPACES TO HOLD-AREA.
000042          MOVE SPACES TO RECORD-TYPE
```

Figure 5-60. Inserted Statements Must Precede the COBOL Statements That Are Skipped

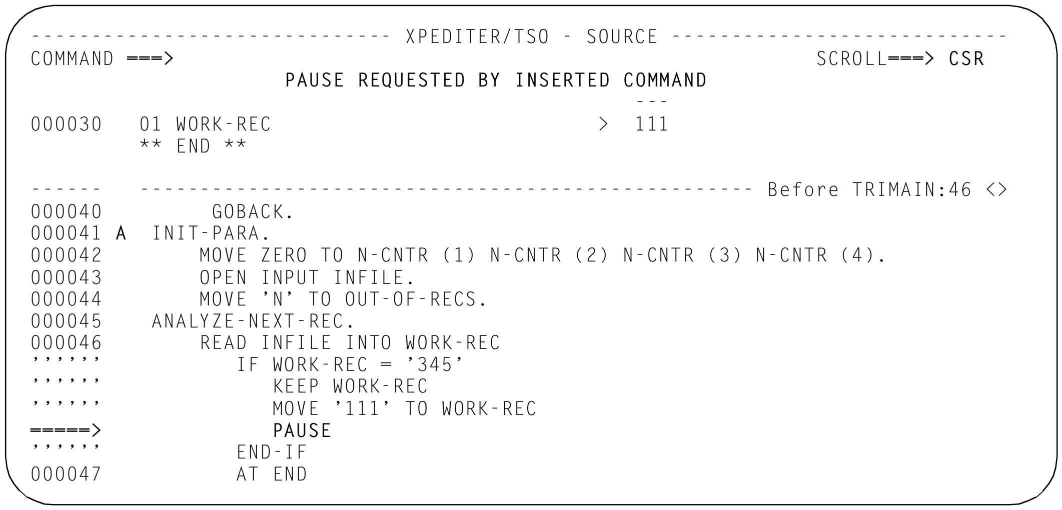
This approach makes it easy to experiment with several potential fixes.

Inserting Statements

You can insert XPEDITER/TSO commands, such as MOVE, PEEK, GOTO, and PAUSE, using the IF...ELSE... constructs to your program. The capability to insert statements allows you to test fixes before you update the source code and actually recompile the program. Inserted statements are executed after the last logical statement as if they are part of the source code. Only one inserted command per line is permitted.

You can also dynamically insert SQL statements and prototype DB2 applications if you have XPEDITER for DB2 Extension and File-AID for DB2 installed. Refer to “Using XPEDITER for DB2 Extension” on page 5-40 for more information.

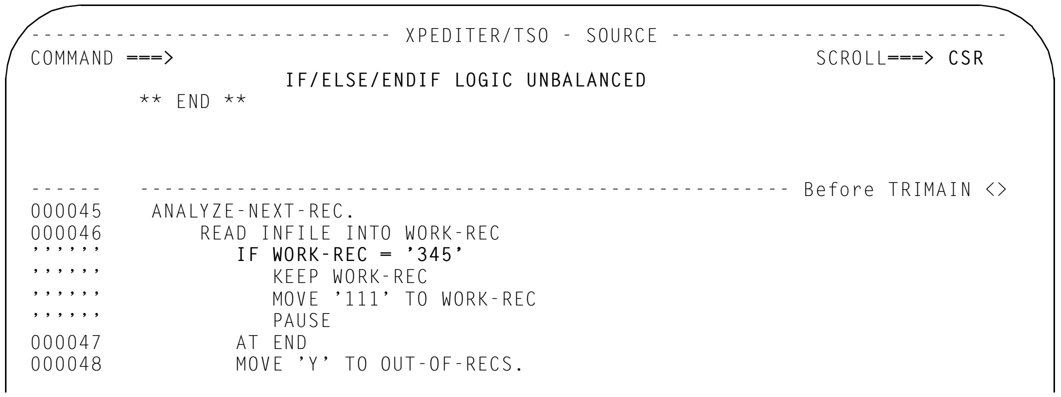
Figure 5-61 on page 5-37 shows the effect of the inserted statements being executed.The PAUSE command can be used to set a breakpoint within a block of inserted XPEDITER/TSO commands or SQL statements. When the pause breakpoint is encountered, XPEDITER/TSO temporarily pauses execution, issues a message, and returns control to you, as shown.



```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                      SCROLL===> CSR
                    PAUSE REQUESTED BY INSERTED COMMAND
                                                           ---
000030   01 WORK-REC                          >  111
         ** END **

------   --------------------------------------------------- Before TRIMAIN:46 <>
000040         GOBACK.
000041 A  INIT-PARA.
000042        MOVE ZERO TO N-CNTR (1) N-CNTR (2) N-CNTR (3) N-CNTR (4).
000043        OPEN INPUT INFILE.
000044        MOVE 'N' TO OUT-OF-RECS.
000045    ANALYZE-NEXT-REC.
000046        READ INFILE INTO WORK-REC
''''''           IF WORK-REC = '345'
''''''              KEEP WORK-REC
''''''              MOVE '111' TO WORK-REC
=====>              PAUSE
''''''           END-IF
000047           AT END
```

Figure 5-61. Result of Executing the Inserted Statements and Encountering the PAUSE Command

The Source display screen is designed after the ISPF/PDF editor.The COBOL source code itself cannot be edited; however, you can insert XPEDITER/TSO commands to the display-only source code by typing over the statement number area with the I (Insert) line command. Use the D (Delete) line command to delete any lines. The syntax of the inserted statements is checked by XPEDITER/TSO before they are executed. If the syntax is incorrect, an error message is generated and the incorrect statement is highlighted when you press **Enter**, as illustrated in Figure 5-62.



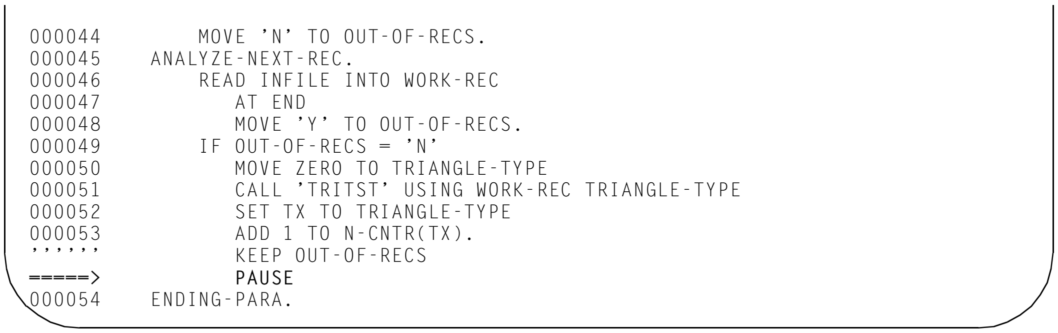
```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                      SCROLL===> CSR
                    IF/ELSE/ENDIF LOGIC UNBALANCED
         ** END **



------   ------------------------------------------------------ Before TRIMAIN <>
000045    ANALYZE-NEXT-REC.
000046        READ INFILE INTO WORK-REC
''''''           IF WORK-REC = '345'
''''''              KEEP WORK-REC
''''''              MOVE '111' TO WORK-REC
''''''              PAUSE
000047           AT END
000048           MOVE 'Y' TO OUT-OF-RECS.
```

Figure 5-62. Syntax Checking for Inserted Statements

INSERT Processing

You can enter the I (Insert) line command on a COBOL statement containing an executable verb (PMAP record) and enter XPEDITER/TSO commands following the statement. Only one inserted command per line is permitted. You cannot insert lines after a statement containing only ELSE, AT END, or scope terminators (END-IF, END-READ, END-PERFORM), where a breakpoint cannot be set. The reason for this rule is that XPEDITER/TSO internally generates an after breakpoint on the COBOL statement where the I line command is entered and interpretively executes the inserted statements only if the internal after breakpoint is reached. In other words, the inserted statements are associated with the COBOL code above them.

When commands are embedded at the end of a conditional structure that is delimited by a period or a scope terminator, the inserted statements will be executed when the COBOL code above them is reached. If you want the inserted statements to be executed only when the true path is taken, place the insert anywhere inside the true path, as shown in Figure 5-63 on page 5-38. If you want the inserted statements to be executed only when the false path is taken, place the insert anywhere inside the false path.



```
000044         MOVE 'N' TO OUT-OF-RECS.
000045     ANALYZE-NEXT-REC.
000046         READ INFILE INTO WORK-REC
000047            AT END
000048            MOVE 'Y' TO OUT-OF-RECS.
000049         IF OUT-OF-RECS = 'N'
000050            MOVE ZERO TO TRIANGLE-TYPE
000051            CALL 'TRITST' USING WORK-REC TRIANGLE-TYPE
000052            SET TX TO TRIANGLE-TYPE
000053            ADD 1 TO N-CNTR(TX).
''''''            KEEP OUT-OF-RECS
=====>            PAUSE
000054     ENDING-PARA.
```

Figure 5-63. Inserting Statements Following a Conditional Construct

Placing the I line command on branching verbs such as PERFORM, GOBACK, and EXIT is not permitted. A message `INSERT NOT PERMITTED FOLLOWING` *verb* is issued when you attempt to do so. If you wish to execute statements following the return from the actual performed paragraph, insert the statement after the last executable code in the out-of-line paragraph.

Refer to the INSERT command in the *XPEDITER/TSO and XPEDITER/IMS Reference Manual* for a list of commands that can and cannot be inserted in your program.

Redirecting Logic

You can dynamically alter the control flow and force the program to take a certain path by using the GOTO command, or by simply changing the data that is processed using the MOVE command and let the program take its own course.

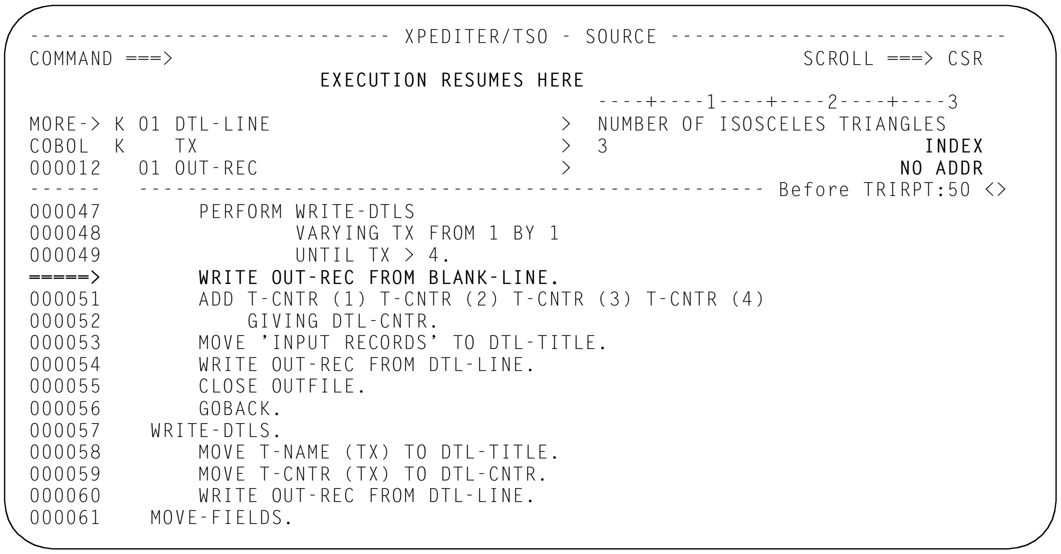
Using the GOTO Command

The GOTO command forces logic changes by redirecting the next executable statement to elsewhere in the program. The command can be used to execute a wild branch, to bypass statements, to test a loop repeatedly, or to take an alternate path. If your module contains nested programs, you cannot use GOTO to branch to another nested program or to go to a separately compiled program. GOTO is restricted to the current program.

Enter the following GOTO command to redirect execution from statement 58 to statement 50 in the middle of a PERFORM VARYING clause, so that this statement never gets completed.

```
GOTO 50
```

Figure 5-64 on page 5-39 shows the result of entering this command.

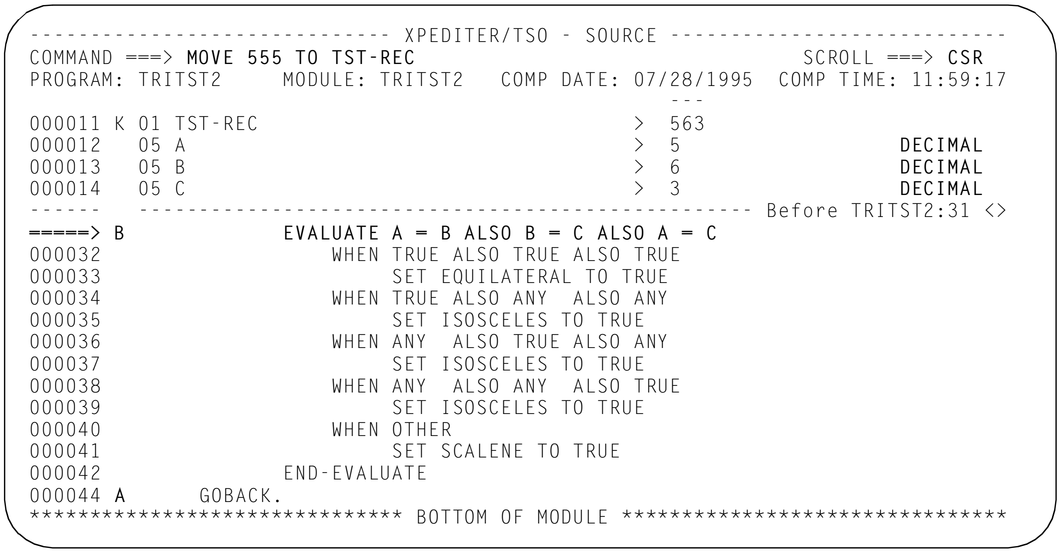


```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                    SCROLL ===> CSR
                        EXECUTION RESUMES HERE
                                                 ----+----1----+----2----+----3
MORE-> K 01 DTL-LINE                         >  NUMBER OF ISOSCELES TRIANGLES
COBOL  K    TX                               >  3                         INDEX
000012   01 OUT-REC                          >                          NO ADDR
------   --------------------------------------------------- Before TRIRPT:50 <>
000047        PERFORM WRITE-DTLS
000048                VARYING TX FROM 1 BY 1
000049                UNTIL TX > 4.
=====>        WRITE OUT-REC FROM BLANK-LINE.
000051        ADD T-CNTR (1) T-CNTR (2) T-CNTR (3) T-CNTR (4)
000052            GIVING DTL-CNTR.
000053        MOVE 'INPUT RECORDS' TO DTL-TITLE.
000054        WRITE OUT-REC FROM DTL-LINE.
000055        CLOSE OUTFILE.
000056        GOBACK.
000057    WRITE-DTLS.
000058        MOVE T-NAME (TX) TO DTL-TITLE.
000059        MOVE T-CNTR (TX) TO DTL-CNTR.
000060        WRITE OUT-REC FROM DTL-LINE.
000061    MOVE-FIELDS.
```

Figure 5-64. Result of Entering the GOTO Command

Using the MOVE Command

The MOVE command, on the other hand, indirectly allows you to change the execution flow by modifying the values of switches, flags, and data that control the path to be taken. Figure 5-65 shows an example of altering TST-REC so that the EVALUATE statement will set the EQUILATERAL switch instead of the SCALENE switch.



```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===> MOVE 555 TO TST-REC                                SCROLL ===> CSR
PROGRAM: TRITST2     MODULE: TRITST2   COMP DATE: 07/28/1995  COMP TIME: 11:59:17
                                                       ---
000011 K 01 TST-REC                            >  563
000012   05 A                                  >  5                     DECIMAL
000013   05 B                                  >  6                     DECIMAL
000014   05 C                                  >  3                     DECIMAL
------   -------------------------------------------------- Before TRITST2:31 <>
=====> B             EVALUATE A = B ALSO B = C ALSO A = C
000032                   WHEN TRUE ALSO TRUE ALSO TRUE
000033                        SET EQUILATERAL TO TRUE
000034                   WHEN TRUE ALSO ANY  ALSO ANY
000035                        SET ISOSCELES TO TRUE
000036                   WHEN ANY  ALSO TRUE ALSO ANY
000037                        SET ISOSCELES TO TRUE
000038                   WHEN ANY  ALSO ANY  ALSO TRUE
000039                        SET ISOSCELES TO TRUE
000040                   WHEN OTHER
000041                        SET SCALENE TO TRUE
000042               END-EVALUATE
000044 A      GOBACK.
****************************** BOTTOM OF MODULE ********************************
```

Figure 5-65. Altering Path by Modifying Data

Examining Files

You can display a list of all the datasets that are allocated during your test session and find out whether they are available to your application program by typing **SHOW ALLOCATE** on the primary command line.All the files that were allocated by processing the file list specified on the test screen are shown at the bottom of the screen, following ddname XOPTIONS. For example, sample program TRIMAIN reads data in INFILE DD and writes a report to OUTFILE DD.

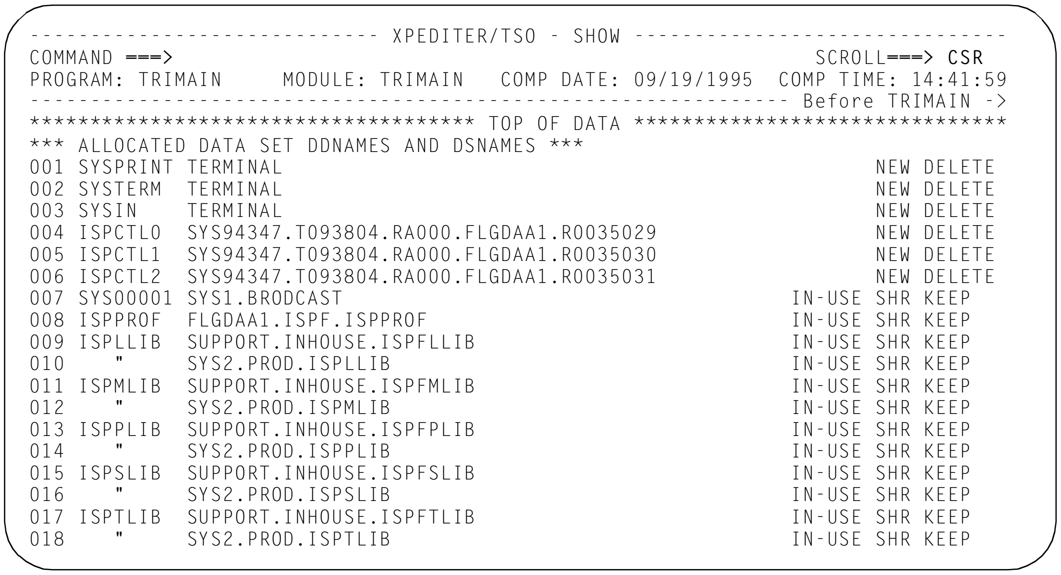
If you discover an open file problem, first check to see if all the necessary files are allocated. If you are missing any files, you can dynamically access the file allocation utility

(FAU) during a test session. Type **ALLOCATE** on the primary command line, allocate the missing files using the file allocation utility, and return to the test session.

You can also execute the TSO ALLOCATE command from the primary command line to allocate additional files.To execute a TSO command, type **TSO** followed by the TSO command; for example:

```
TSO ALLOCATE DD(OUTFILE) DSN(*)
```

Note: Additional files allocated during a test session are not freed when you terminate the test session and return to the test screen. XPEDITER/TSO only allocates and deallocates files that are contained in the file list specified on the test screen. Figure 5-66 contains an example of a SHOW ALLOCATE screen.



```
----------------------------- XPEDITER/TSO - SHOW ------------------------------
COMMAND ===>                                                   SCROLL===> CSR
PROGRAM: TRIMAIN     MODULE: TRIMAIN   COMP DATE: 09/19/1995  COMP TIME: 14:41:59
------------------------------------------------------------- Before TRIMAIN ->
********************************** TOP OF DATA ******************************
*** ALLOCATED DATA SET DDNAMES AND DSNAMES ***
001 SYSPRINT TERMINAL                                                NEW DELETE
002 SYSTERM  TERMINAL                                                NEW DELETE
003 SYSIN    TERMINAL                                                NEW DELETE
004 ISPCTL0  SYS94347.T093804.RA000.FLGDAA1.R0035029                 NEW DELETE
005 ISPCTL1  SYS94347.T093804.RA000.FLGDAA1.R0035030                 NEW DELETE
006 ISPCTL2  SYS94347.T093804.RA000.FLGDAA1.R0035031                 NEW DELETE
007 SYS00001 SYS1.BRODCAST                                    IN-USE SHR KEEP
008 ISPPROF  FLGDAA1.ISPF.ISPPROF                             IN-USE SHR KEEP
009 ISPLLIB  SUPPORT.INHOUSE.ISPFLLIB                         IN-USE SHR KEEP
010    "     SYS2.PROD.ISPLLIB                                IN-USE SHR KEEP
011 ISPMLIB  SUPPORT.INHOUSE.ISPFMLIB                         IN-USE SHR KEEP
012    "     SYS2.PROD.ISPMLIB                                IN-USE SHR KEEP
013 ISPPLIB  SUPPORT.INHOUSE.ISPFPLIB                         IN-USE SHR KEEP
014    "     SYS2.PROD.ISPPLIB                                IN-USE SHR KEEP
015 ISPSLIB  SUPPORT.INHOUSE.ISPFSLIB                         IN-USE SHR KEEP
016    "     SYS2.PROD.ISPSLIB                                IN-USE SHR KEEP
017 ISPTLIB  SUPPORT.INHOUSE.ISPFTLIB                         IN-USE SHR KEEP
018    "     SYS2.PROD.ISPTLIB                                IN-USE SHR KEEP
```

Figure 5-66. SHOW ALLOCATE Screen

You can view the file contents by typing **BROWSE** *ddname* or **BROWSE** *'dataset name'* if the file is a sequential file or a member of a PDS. ISPF allows you to split screens and browse or edit a file while you are debugging your program with XPEDITER/TSO Remember that the XPEDITER/TSO test session default for PF2 is PEEK CSR (not SPLIT) and for PF9 is GO 1 (not SWAP).

None of the ISPF commands are available in the dialog environment since XPEDITER/TSO does not run as a dialog program.

Using XPEDITER for DB2 Extension

This section describes how to:

- Browse and edit DB2 table data while testing your program.
- Analyze SQL statement execution with the FADB2 EXPLAIN command.
- Prototype SQL logic by inserting SQL statements.

Refer to the *File-AID for DB2 Reference Manual* for information about using File-AID for DB2.

Notes:

1. XPEDITER/TSO supports debugging of any DB2 program within all execution environments selectable under the Environments Menu, with the following exceptions:

- The first exception to this XPEDITER/TSO support rule is the XPEDITER/TSO dialog environment.
- The second exception to normal debugging is that XPEDITER for DB2 Extension cannot be accessed when using the batch connect facility.

Browsing and Editing DB2 Table Data

When debugging DB2 programs, you can access File-AID for DB2 to inspect and manipulate the DB2 table data associated with SQL statements being executed.

For example, you can experiment with program SQL statements and then enter **FADB2 EDIT** to access File-AID for DB2 and change the test data associated with the statements.

The FADB2 EDIT command transfers control directly to the File-AID for DB2 Edit function. Once in File-AID for DB2 Edit, you can enter the information to access the table data you want to view. Refer to the *File-AID for DB2 Reference Manual* for information about using the Browse and Edit functions.

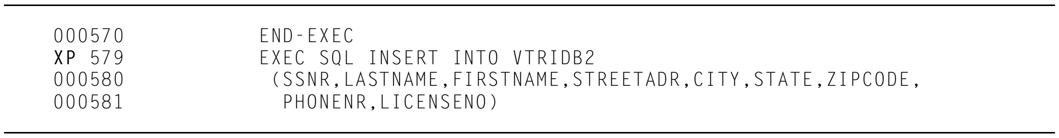
Analyzing SQL Statement Execution with the FADB2 EXPLAIN Command

The FADB2 EXPLAIN command (XP line command) can be entered for any SELECT, DELETE, INSERT, or UPDATE SQL statement in your program to display information about the execution of the statement.

The Explain data is also stored in a DB2 table called *user id*.Plan_Table.The stored Explain data can be used to generate reports using the File-AID for DB2 Reports facility or view the Plan_Table using the File-AID for DB2 Browse facility.

Note: The Plan_Table must exist before you use the FADB2 EXPLAIN command.

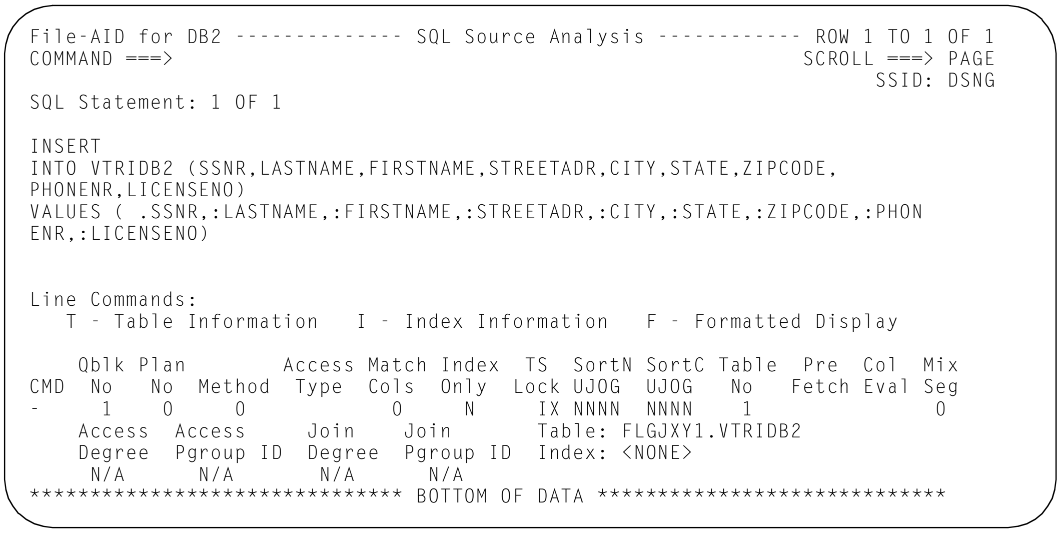
For example, in Figure 5-67, the XP line command is entered on the EXEC SQL INSERT statement number 579.



```
000570          END-EXEC
XP 579          EXEC SQL INSERT INTO VTRIDB2
000580           (SSNR,LASTNAME,FIRSTNAME,STREETADR,CITY,STATE,ZIPCODE,
000581            PHONENR,LICENSENO)
```

Figure 5-67. Using the FADB2 EXPLAIN Line Command

When the FADB2 EXPLAIN command is entered, File-AID for DB2 is accessed, and the SQL Source Analysis screen shown in Figure 5-55 is displayed.



```
File-AID for DB2 ------------- SQL Source Analysis ----------- ROW 1 TO 1 OF 1
COMMAND ===>                                                 SCROLL ===> PAGE
                                                                   SSID: DSNG
SQL Statement: 1 OF 1

INSERT
INTO VTRIDB2 (SSNR,LASTNAME,FIRSTNAME,STREETADR,CITY,STATE,ZIPCODE,
PHONENR,LICENSENO)
VALUES ( .SSNR,:LASTNAME,:FIRSTNAME,:STREETADR,:CITY,:STATE,:ZIPCODE,:PHON
ENR,:LICENSENO)


Line Commands:
   T - Table Information   I - Index Information   F - Formatted Display

    Qblk Plan        Access Match Index  TS  SortN SortC Table  Pre  Col  Mix
CMD  No   No  Method  Type  Cols  Only  Lock UJOG  UJOG   No   Fetch Eval Seg
-     1    0     0           0     N     IX NNNN  NNNN    1               0
    Access  Access     Join    Join       Table: FLGJXY1.VTRIDB2
    Degree  Pgroup ID  Degree  Pgroup ID  Index: <NONE>
     N/A      N/A       N/A      N/A
****************************** BOTTOM OF DATA *****************************
```

Figure 5-68. SQL Source Analysis Screen

Inserting Program SQL Statements

You can dynamically insert and execute SQL statements from within your XPEDITER/TSO source. SQL statements are inserted with the INSERT command. Refer to "Modifying Program Logic" on page 5-28 for detailed information about the INSERT command.

The inserted statements are executed after the last logical statement as if they are part of the source code. The capability to insert SQL statements lets you test fixes before you update the source code and recompile the program.

Each inserted SQL statement **must** be prefixed by EXEC SQL and suffixed by END-EXEC. Otherwise, XPEDITER/TSO issues a syntax error message. Any data entered after the END-EXEC statement, on the same line, is ignored. Error messages regarding inserted SQL statements are displayed in the same manner as error messages regarding other inserted statements. Additional help is available for negative SQL return codes by entering the primary command FADB2 HELP after the error message is displayed.

SQL statements are allowed within an inserted IF construct. For example,

```
IF A = B
   EXEC SQL
     SELECT
   END-EXEC
   MOVE 1 TO A
   EXEC SQL
     SELECT
   END-EXEC
END-IF.
```

Inserted SQL statements appear in the XPEDITER/TSO log and script in the same manner as other inserted statements. You can keep or display host variables in inserted SQL statements.

SQL Statements That Can Be Inserted in Your Program

The following SQL statements are supported (i.e., the statements can be inserted in your source and are valid in a debugging session):

ALTER INDEX	Changes the description of an index.
ALTER STOGROUP	Changes the description of a storage group.

ALTER TABLE	Changes the description of a table.
ALTER TABLESPACE	Changes the description of a table space.
BEGIN DECLARE	Marks the beginning of a host variable declaration section.
CLOSE	Closes the cursor and deletes the temporary application-specific result table.
COMMENT ON	Replaces or adds a comment to the description of a table, view, or column.
COMMIT	Terminates a unit of recovery and commits the DB2 table changes made by that unit of recovery.
CREATE DATABASE	Defines a database.
CREATE INDEX	Creates an index on a table.
CREATE STOGROUP	Defines a storage group or set of volumes, controlled by a VSAM catalog, on which storage can later be allocated for table spaces and indexes.
CREATE SYNONYM	Defines an alternate name for a table or view.
CREATE TABLE	Creates a table.
CREATE TABLESPACE	Allocates and formats table spaces.
CREATE VIEW	Defines a view of one or more tables.
DECLARE CURSOR	Associates a cursor name with OPEN, FETCH, and CLOSE statements, which declare and retrieve data from an application specific result table row-by-row. DECLARE CURSOR can be inserted into the executable portion of a program.
DECLARE STATEMENT	Declares a statement for dynamic SQL.
DECLARE TABLE	Declares a table. The DECLARE TABLE declarative statement can be inserted into the application program. It causes host variable locations to be defined in accordance to the columns of the declared table. You can reference these columns in subsequent SQL statements.
DELETE	Deletes one or more rows from a table.
DESCRIBE	Provides a description of the columns in a table or view.
DROP	Removes an object and its description in the DB2 catalog.
END DECLARE	Marks the ending of a host variable declaration section.
EXECUTE	Executes a prepared SQL statement.
EXECUTE IMMEDIATE	Prepares and executes an SQL statement.
EXPLAIN	Obtains information about how an SQL statement will be executed. An example showing the use of EXPLAIN is given in "Analyzing SQL Statement Execution" on page 5-32.
FETCH	Positions the cursor on the next row of the application-specific result table and assigns the values of that row to host variables in the application program.
GRANT	Grants privileges.
INSERT	Inserts rows into a table or view.
LABEL ON	Adds or replaces labels in the catalog descriptions of tables, views, columns, or sets of columns.
LOCK TABLE	Acquires a shared or exclusive lock on a table.

OPEN	Opens a cursor so that it can be used by FETCH to fetch rows from the application-specific result table.
PREPARE	Dynamically prepares an SQL statement for execution.
REVOKE	Revokes privileges.
ROLLBACK	Terminates a unit of recovery and backs out database changes made by that unit of recovery.
SELECT	Specifies a result table and selects rows to view.
SET	Changes the value of the authorization ID.
UPDATE	Updates the values of specified columns in rows of a table or view.

Notes:

1. The syntax for all SQL statements is explained in the *IBM Database 2 Reference* manual.
2. Host variables can be used within inserted SQL. XPEDITER/TSO requires that each host variable be preceded by a colon even though DB2 does not always require it.
3. Within the BTS/DLI setup, recovery of DB2 tables and IMS databases is uncoordinated. The SQL COMMIT and ROLLBACK commands commit and rollback changes made to DB2 tables only; they do not affect your IMS databases.

Expanding EXEC SQL and EXEC CICS Statements

The GEN command can be used to enable the debugging of EXEC SQL, EXEC DLI, and EXEC CICS statements. The GEN command lets you expand EXEC statements and display translator-related statements. You can see and place breakpoints on the translated code and debug the EXECs while the program is being executed.

If SET GEN is OFF (the system default), the unexpanded code is displayed. The GEN command can be used to selectively expand and display translated code. The SET GEN ON command globally expands and displays all statements. If SET GEN is ON, there is no requirement to use the GEN command since all EXECs have already been expanded. SET GEN ON remains operational across debugging sessions.

If GEN is entered as a primary command, the statement number must point to the start of the EXEC statement. You can specify a list or a range of statement numbers.

When a GEN is entered, the EXEC appears on the Source display screen as code that is commented out. The expanded code generated by the translator is also displayed. Any breakpoints that have been set on the EXEC statement appear in the expanded code. After expansion, breakpoints on the commented code are not allowed, but breakpoints on the expanded code are allowed.

On a TRACE or WHEN breakpoint, each statement within the expanded code receives the appropriate breakpoint.

On a COUNT, the expansion gives counts for all statements. When collapsed, the count on the first statement is moved to the EXEC statement.

An INSERT on an EXEC is moved after the last statement in the expanded code. If you insert on expanded code, the inserted data is lost when the expanded code is collapsed.

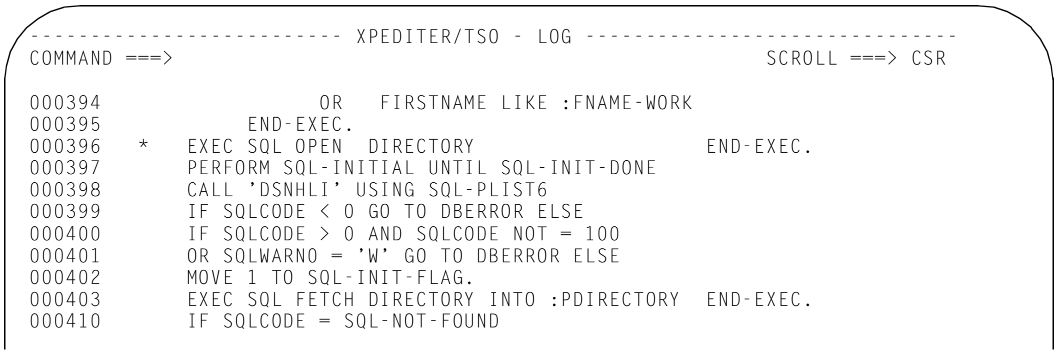
The G line command must be entered on the line at the start of the EXEC statement. The block command, GG, must begin and end on the first lines of EXEC statements. The line command, DG, removes the effects of the G or GG line commands, as does the DELETE GEN primary command.

Example 1—Expanding an EXEC SQL Statement Using a G Line Command:

As seen in Figure 5-69, the G line command is entered on statement number 396, which results in the expanded text shown in Figure 5-70.

```
000394                OR   FIRSTNAME LIKE :FNAME-WORK
000395          END-EXEC.
G  396       EXEC SQL OPEN  DIRECTORY                   END-EXEC.
000403       EXEC SQL FETCH DIRECTORY INTO :PDIRECTORY  END-EXEC.
000410       IF SQLCODE = SQL-NOT-FOUND
```

Figure 5-69. Expanding an EXEC SQL Statement Using the G(en) Line Command



```
------------------------- XPEDITER/TSO - LOG ------------------------------
COMMAND ===>                                                  SCROLL ===> CSR

000394                OR   FIRSTNAME LIKE :FNAME-WORK
000395          END-EXEC.
000396   *   EXEC SQL OPEN  DIRECTORY                   END-EXEC.
000397       PERFORM SQL-INITIAL UNTIL SQL-INIT-DONE
000398       CALL 'DSNHLI' USING SQL-PLIST6
000399       IF SQLCODE < 0 GO TO DBERROR ELSE
000400       IF SQLCODE > 0 AND SQLCODE NOT = 100
000401       OR SQLWARN0 = 'W' GO TO DBERROR ELSE
000402       MOVE 1 TO SQL-INIT-FLAG.
000403       EXEC SQL FETCH DIRECTORY INTO :PDIRECTORY  END-EXEC.
000410       IF SQLCODE = SQL-NOT-FOUND
```

Figure 5-70. Expanded EXEC SQL Statement

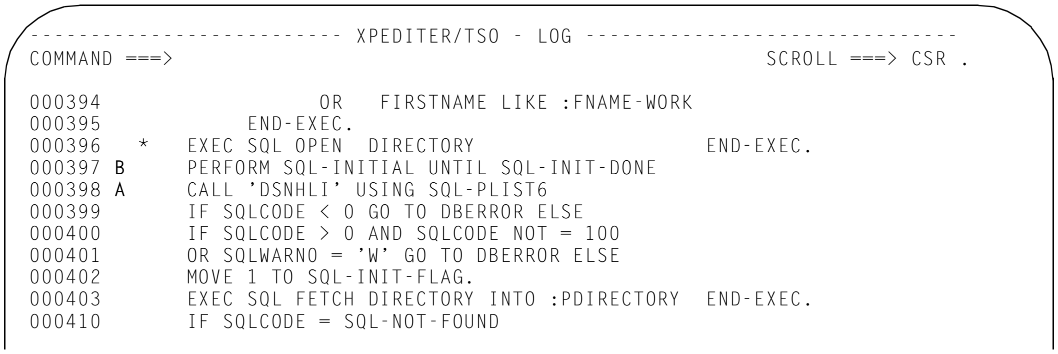
Example 2—Seeing the Effect of Breakpoints Set Against an EXEC:

In Figure 5-71, note that before and after breakpoints have been entered on the EXEC SQL statement number 396, as denoted by the @ sign. If you gen this EXEC statement, you can see where the breakpoints are applied in the translated code, as seen in Figure 5-72.

```
000394                OR   FIRSTNAME LIKE :FNAME-WORK
000395          END-EXEC.
G  396 @     EXEC SQL OPEN  DIRECTORY                   END-EXEC.
000403       EXEC SQL FETCH DIRECTORY INTO :PDIRECTORY  END-EXEC.
000410       IF SQLCODE = SQL-NOT-FOUND
```

Figure 5-71. Before and After Breakpoints Set on an Unexpanded EXEC SQL Statement

After expansion, a before breakpoint on the EXEC SQL appears on the first verb of the expanded code and an after breakpoint appears on the call to DSNHLI.



```
------------------------- XPEDITER/TSO - LOG ------------------------------
COMMAND ===>                                                  SCROLL ===> CSR .

000394                OR   FIRSTNAME LIKE :FNAME-WORK
000395          END-EXEC.
000396   *   EXEC SQL OPEN  DIRECTORY                   END-EXEC.
000397 B     PERFORM SQL-INITIAL UNTIL SQL-INIT-DONE
000398 A     CALL 'DSNHLI' USING SQL-PLIST6
000399       IF SQLCODE < 0 GO TO DBERROR ELSE
000400       IF SQLCODE > 0 AND SQLCODE NOT = 100
000401       OR SQLWARN0 = 'W' GO TO DBERROR ELSE
000402       MOVE 1 TO SQL-INIT-FLAG.
000403       EXEC SQL FETCH DIRECTORY INTO :PDIRECTORY  END-EXEC.
000410       IF SQLCODE = SQL-NOT-FOUND
```

Figure 5-72. Before and After Breakpoints Shown on a Genned EXEC SQL Statement

Debugging a Sourceless Program

XPEDITER/TSO allows you to interactively debug programs that do not have the source (no source listing in the DDIO) available, such as old modules and third party packages.

With this support, you can debug "sourceless" main programs or subprograms in the same manner as interactive source level debugging, with a few differences. Test session setup and startup is the same as setting up a session to interactively debug at the source level. The differences occur during the debugging session:

- XPEDITER/TSO debugging commands such as AFTER, BEFORE, COUNT, SKIP, TRACE, WHEN, PEEK, KEEP, GOTO, and DELETE cannot be used.
- To set a breakpoint, you must use the AT command, which sets a before breakpoint.

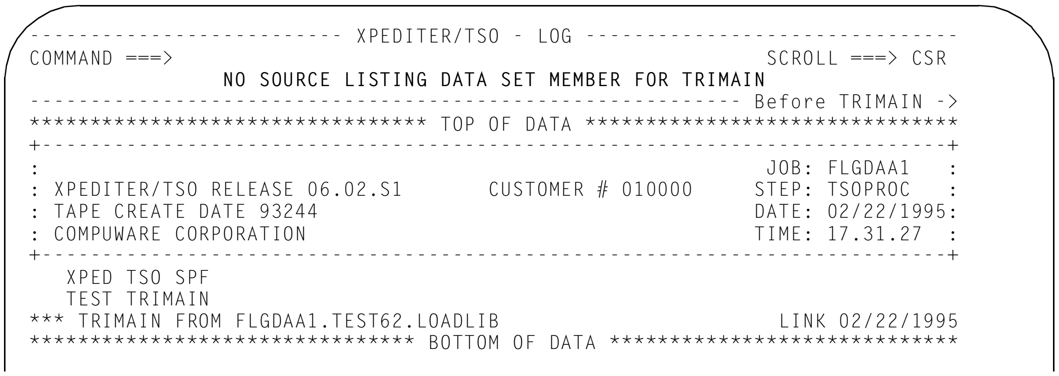
  When the AT breakpoint is reached, the AT Display screen shows your sourceless program in dump format. You can enter XPEDITER/TSO debugging commands such as GPREGS, MEMORY, SHOW, and so on.

Accessing a Sourceless Main Program

When program execution begins and the main program **does not** have source, the log is automatically displayed with the message,

```
NO SOURCE LISTING DATA SET MEMBER FOR module-name.
```

Refer to Figure 5-56.



```
------------------------- XPEDITER/TSO - LOG ------------------------------
COMMAND ===>                                               SCROLL ===> CSR
                NO SOURCE LISTING DATA SET MEMBER FOR TRIMAIN
---------------------------------------------------------- Before TRIMAIN ->
********************************* TOP OF DATA ******************************
+--------------------------------------------------------------------------+
:                                                            JOB: FLGDAA1   :
: XPEDITER/TSO RELEASE 06.02.S1       CUSTOMER # 010000     STEP: TSOPROC   :
: TAPE CREATE DATE 93244                                    DATE: 02/22/1995:
: COMPUWARE CORPORATION                                     TIME: 17.31.27  :
+--------------------------------------------------------------------------+
   XPED TSO SPF
   TEST TRIMAIN
*** TRIMAIN FROM FLGDAA1.TEST62.LOADLIB                       LINK 02/22/1995
******************************** BOTTOM OF DATA ****************************
```

Figure 5-73. Log Showing No Source Message for a Driver Program

You can use the AT command on the Log screen to set a breakpoint in the specified program if you are familiar with the program. However, if you are not familiar with the program, you can use the MEMORY command to display the program in dump format. For example:

```
MEMORY TRIMAIN:
```

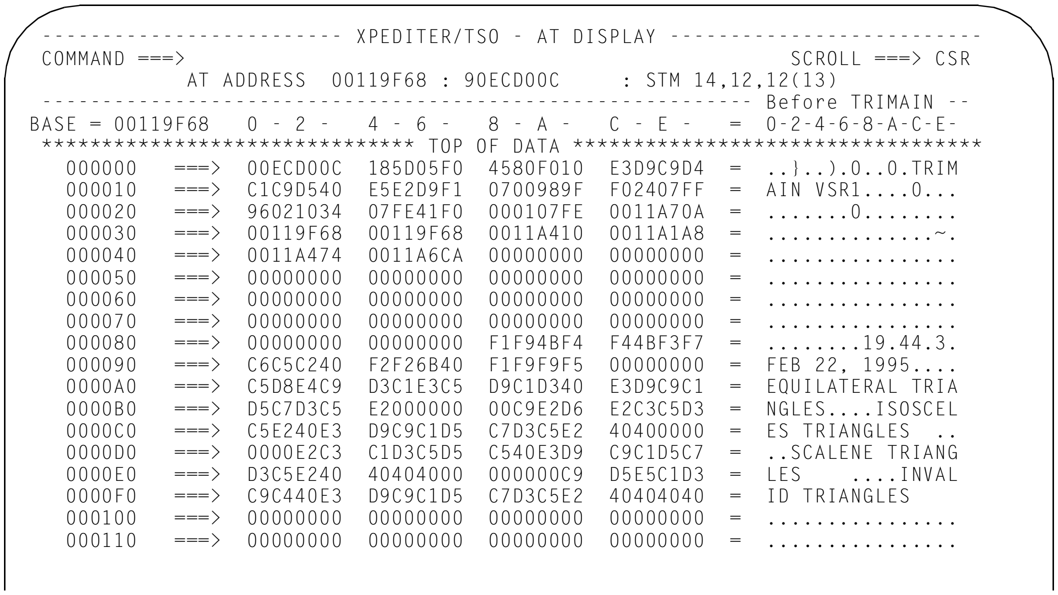
displays the Memory screen showing a dump of TRIMAIN.

The AT command can then be entered to set breakpoints in the program. Issuing the AT command will also load the module into memory if it is not loaded. The full syntax of the AT command is provided in the *XPEDITER/TSO and XPEDITER/IMS Reference Manual*.

Note: The AT command can also be entered in an initial script.

For example, entering the following on the Memory screen displays the At Display screen shown in Figure 5-57.

```
AT TRIMAIN:;GO
```



```
------------------------ XPEDITER/TSO - AT DISPLAY ------------------------
COMMAND ===>                                                SCROLL ===> CSR
            AT ADDRESS  00119F68 : 90ECD00C     : STM 14,12,12(13)
------------------------------------------------------------ Before TRIMAIN --
BASE = 00119F68   0 - 2 -   4 - 6 -   8 - A -   C - E -   =  0-2-4-6-8-A-C-E-
****************************** TOP OF DATA *********************************
   000000   ===>  00ECD00C  185D05F0  4580F010  E3D9C9D4  =  ..}..).0..0.TRIM
   000010   ===>  C1C9D540  E5E2D9F1  0700989F  F02407FF  =  AIN VSR1....0...
   000020   ===>  96021034  07FE41F0  000107FE  0011A70A  =  .......0........
   000030   ===>  00119F68  00119F68  0011A410  0011A1A8  =  ..............~.
   000040   ===>  0011A474  0011A6CA  00000000  00000000  =  ................
   000050   ===>  00000000  00000000  00000000  00000000  =  ................
   000060   ===>  00000000  00000000  00000000  00000000  =  ................
   000070   ===>  00000000  00000000  00000000  00000000  =  ................
   000080   ===>  00000000  00000000  F1F94BF4  F44BF3F7  =  ........19.44.3.
   000090   ===>  C6C5C240  F2F26B40  F1F9F9F5  00000000  =  FEB 22, 1995....
   0000A0   ===>  C5D8E4C9  D3C1E3C5  D9C1D340  E3D9C9C1  =  EQUILATERAL TRIA
   0000B0   ===>  D5C7D3C5  E2000000  00C9E2D6  E2C3C5D3  =  NGLES....ISOSCEL
   0000C0   ===>  C5E240E3  D9C9C1D5  C7D3C5E2  40400000  =  ES TRIANGLES  ..
   0000D0   ===>  0000E2C3  C1D3C5D5  C540E3D9  C9C1D5C7  =  ..SCALENE TRIANG
   0000E0   ===>  D3C5E240  40404000  000000C9  D5E5C1D3  =  LES    ....INVAL
   0000F0   ===>  C9C440E3  D9C9C1D5  C7D3C5E2  40404040  =  ID TRIANGLES
   000100   ===>  00000000  00000000  00000000  00000000  =  ................
   000110   ===>  00000000  00000000  00000000  00000000  =  ................
```

Figure 5-74. AT Display Screen

If the address specified with the AT command is not a valid instruction (data or privileged), a message is returned. Also, if the specified program has source, no breakpoint is set and the message `MODULE HAS SOURCE; USE XPEDITER BREAKPOINT COMMANDS` is displayed.

On the AT Display screen, the offset and the instruction pointer are highlighted, indicating where the breakpoint is set. The message area contains the breakpoint address and the instruction in hexadecimal and mnemonic form. If the AT command is used on an address and not in a module loaded by XPEDITER/TSO, the compile date and time displayed on the third line will be blank.

Accessing a Sourceless Subprogram

When your debugging session begins and your main program has source, the Source display screen is displayed. You can set breakpoints in the regular manner on this screen.

Normally, breakpoints can be set in a subprogram by entering a fully qualified breakpoint command or using the SOURCE or INTERCEPT commands to access the subprogram from this screen. However, if the subprogram does not have source, the message `NO SOURCE LISTING INFORMATION FOUND FOR MODULE` is displayed at the top of the source display.

If you are not familiar with the program, you can use the MEMORY command to access main storage and display the module in dump format on the Memory screen. For example:

```
MEMORY TRIMAIN:
```

With the program displayed on the Memory screen, you can decide exactly where you want to set breakpoints using the AT command.

If you are familiar with the application you are debugging, you can use the AT command with the module/procedure name to set a breakpoint at the 0 displacement in the module. For example:

```
AT TRIMAIN:
```

When the AT breakpoint is encountered, the AT Display screen shown in Figure 5-57 on page 5-36 is displayed and program execution is paused at the specified offset.

Using XPEDITER/TSO Commands for Sourceless Debugging

The rules for using XPEDITER/TSO commands for sourceless debugging are as follows:

Primary Commands

None of the debugging commands, such as, AFTER, BEFORE, COUNT, SKIP, TRACE, WHEN, PEEK, KEEP, GOTO, GO *n*, and DELETE are valid. The GO command is valid. All other commands can be used.

Line Commands

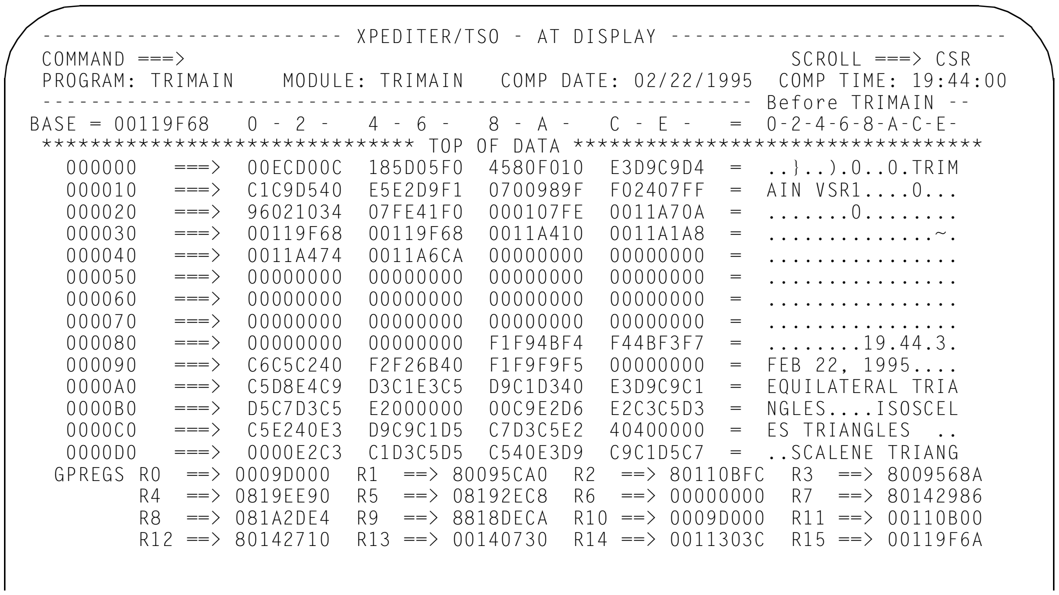
The only valid line commands are X (eXclude) and T (Template). To insure proper results, the rest of the line command area should be blanked out after entering the line command.

Note: Typing over instructions with X'00A3' will not be recognized as a breakpoint. A S0C1 abend occurs when the typed over instruction is executed.

The following are examples of using some XPEDITER/TSO commands to debug a sourceless program.

Using the UP command on the AT Display screen scrolls the screen toward the beginning of the module/procedure. Scrolling beyond the beginning of the module/procedure displays negative offsets. Using the LOCATE * command scrolls to the current execution point.

The GPREGS command can be used to open a modifiable window at the bottom of your screen. The register values displayed can be changed by typing over the values. An example of the GPREGS window is shown in Figure 5-58.

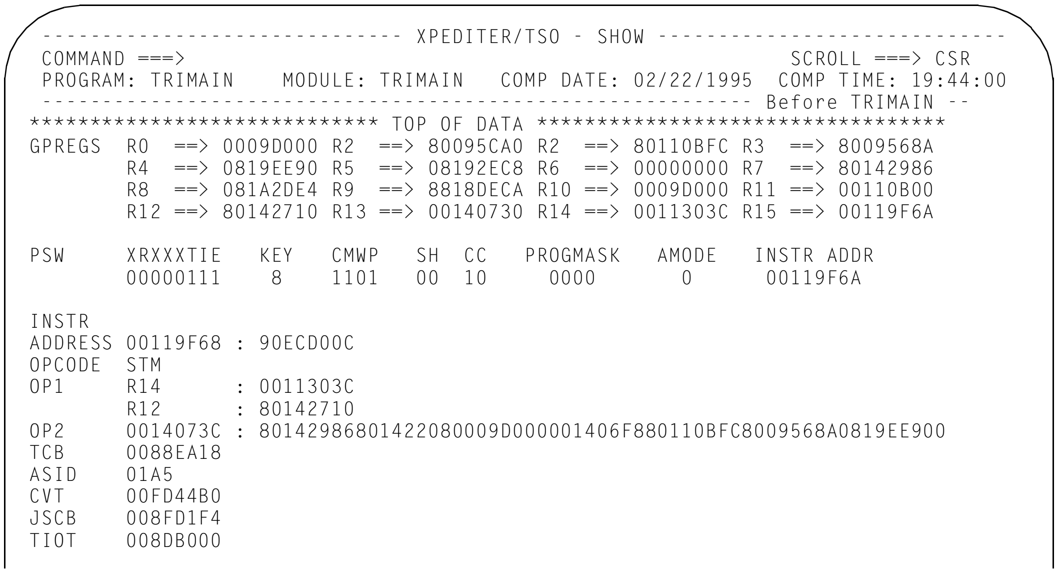


```
 ------------------------- XPEDITER/TSO - AT DISPLAY ---------------------------
 COMMAND ===>                                                     SCROLL ===> CSR
 PROGRAM: TRIMAIN    MODULE: TRIMAIN   COMP DATE: 02/22/1995  COMP TIME: 19:44:00
 --------------------------------------------------------- Before TRIMAIN --
BASE = 00119F68   0 - 2 -   4 - 6 -   8 - A -   C - E -   =  0-2-4-6-8-A-C-E-
 ****************************** TOP OF DATA *********************************
   000000   ===>  00ECD00C  185D05F0  4580F010  E3D9C9D4  =  ..}..).0..0.TRIM
   000010   ===>  C1C9D540  E5E2D9F1  0700989F  F02407FF  =  AIN VSR1....0...
   000020   ===>  96021034  07FE41F0  000107FE  0011A70A  =  .......0........
   000030   ===>  00119F68  00119F68  0011A410  0011A1A8  =  ..............~.
   000040   ===>  0011A474  0011A6CA  00000000  00000000  =  ................
   000050   ===>  00000000  00000000  00000000  00000000  =  ................
   000060   ===>  00000000  00000000  00000000  00000000  =  ................
   000070   ===>  00000000  00000000  00000000  00000000  =  ................
   000080   ===>  00000000  00000000  F1F94BF4  F44BF3F7  =  ........19.44.3.
   000090   ===>  C6C5C240  F2F26B40  F1F9F9F5  00000000  =  FEB 22, 1995....
   0000A0   ===>  C5D8E4C9  D3C1E3C5  D9C1D340  E3D9C9C1  =  EQUILATERAL TRIA
   0000B0   ===>  D5C7D3C5  E2000000  00C9E2D6  E2C3C5D3  =  NGLES....ISOSCEL
   0000C0   ===>  C5E240E3  D9C9C1D5  C7D3C5E2  40400000  =  ES TRIANGLES  ..
   0000D0   ===>  0000E2C3  C1D3C5D5  C540E3D9  C9C1D5C7  =  ..SCALENE TRIANG
  GPREGS R0  ==> 0009D000  R1  ==> 80095CA0  R2  ==> 80110BFC  R3  ==> 8009568A
         R4  ==> 0819EE90  R5  ==> 08192EC8  R6  ==> 00000000  R7  ==> 80142986
         R8  ==> 081A2DE4  R9  ==> 8818DECA  R10 ==> 0009D000  R11 ==> 00110B00
         R12 ==> 80142710  R13 ==> 00140730  R14 ==> 0011303C  R15 ==> 00119F6A
```

Figure 5-75. GPREGS Window on the AT Display Screen

Data and instruction areas can be typed over, as long as a breakpoint is not set on the instruction being typed over.

The SHOW ACTIVE command can be entered to display a summary of general-purpose register contents, PSW, and some control block information. The result of the SHOW ACTIVE command is shown in Figure 5-59.



```
 ------------------------------ XPEDITER/TSO - SHOW ----------------------------
 COMMAND ===>                                                  SCROLL ===> CSR
 PROGRAM: TRIMAIN    MODULE: TRIMAIN   COMP DATE: 02/22/1995  COMP TIME: 19:44:00
 -------------------------------------------------------------- Before TRIMAIN --
***************************** TOP OF DATA **********************************
GPREGS  R0  ==> 0009D000 R2  ==> 80095CA0 R2  ==> 80110BFC R3  ==> 8009568A
        R4  ==> 0819EE90 R5  ==> 08192EC8 R6  ==> 00000000 R7  ==> 80142986
        R8  ==> 081A2DE4 R9  ==> 8818DECA R10 ==> 0009D000 R11 ==> 00110B00
        R12 ==> 80142710 R13 ==> 00140730 R14 ==> 0011303C R15 ==> 00119F6A

PSW     XRXXXTIE   KEY   CMWP   SH  CC    PROGMASK   AMODE   INSTR ADDR
        00000111    8    1101   00  10      0000       0      00119F6A

INSTR
ADDRESS 00119F68 : 90ECD00C
OPCODE  STM
OP1     R14      : 0011303C
        R12      : 80142710
OP2     0014073C : 801429868014220800009D000001406F880110BFC8009568A0819EE900
TCB     0088EA18
ASID    01A5
CVT     00FD44B0
JSCB    008FD1F4
TIOT    008DB000
```

Figure 5-76. SHOW ACTIVE Display Screen

The END command returns you to the previously displayed AT Display screen as shown in Figure 5-58 on page 5-37.

The SHOW AT command can also be entered to display all the outstanding breakpoints set by the AT command.

Use the DELETE AT command to remove all outstanding AT breakpoints.

Displaying Environmental Data

If you have Abend-AID release 7.0.2 or above installed at your site, you can use the AA SNAP command to display an Abend-AID Snapshot report containing environment specific run-time characteristics during a test session. If you have the VSAM, IDMS, IMS, or DB2 Abend-AID options, the report displays subsystem-related debugging information as follows:

IDMS

- CA-IDMS environment data (subschema, commit point, status)
- Subschema control (DB-Key information, current and error record/area)
- Database command trace (database call and status trace)
- DB-Key cross reference
- Current records (record name, DB-Key)
- Current sets (set name, record name, program reference, DB-Key)
- Current areas (area name, record name, mode, program reference, DB-Key)

DB2

- SQL return code
- Host variable
- SQL statement
- DB2 release, subsystem, authorization, attach mode

- Plan, bind date/time, isolation, acquire, release, validate
- DBRM, precompile date/time, host language, SQL escape, SQL decimal
- Host variables referenced
- Table and column definition
- SQLCA
- Plan dependencies

IMS

- Function call
- Current PCB (PCB address, database name, segment level, status code, process options, segment name, Key length, number of segments, Key feedback length)
- JCB database call trace (call type, status code, description)
- SSA

VSAM

- Dataset ddname
- Access method
- Record and Key length
- File request type

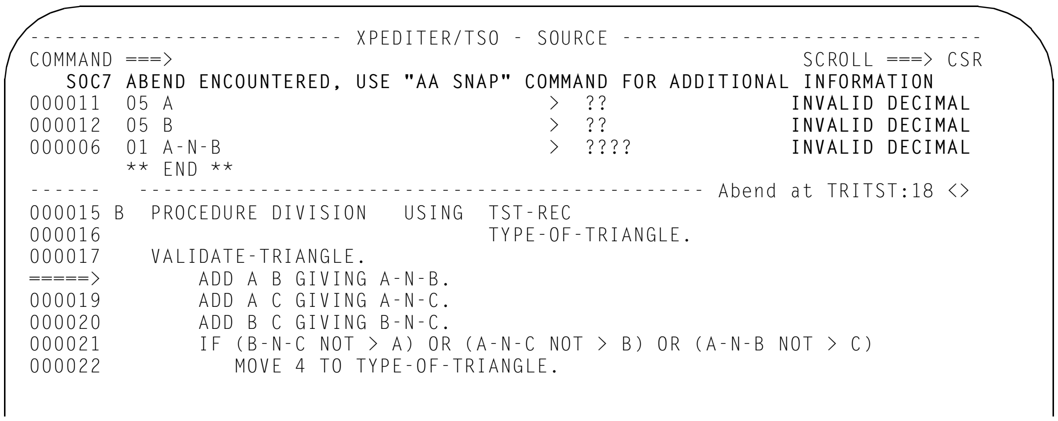
For an example of displaying a Snapshot report and browsing the data, refer to "Displaying the Abend-AID Snapshot Report" on page 6-1.

Chapter 6. Handling Run-Time Errors

XPEDITER/TSO intercepts program abends and prevents the system from producing a dump.

When an abend occurs in a program for which the source is available, execution automatically pauses, the execution arrow points to the offending statement, and a message indicating the action to take is displayed.

Figure 6-1 is an example of XPEDITER/TSO responding to an abend.



```
------------------------- XPEDITER/TSO - SOURCE -----------------------------
COMMAND ===>                                                   SCROLL ===> CSR
   S0C7 ABEND ENCOUNTERED, USE "AA SNAP" COMMAND FOR ADDITIONAL INFORMATION
000011  05 A                               >  ??              INVALID DECIMAL
000012  05 B                               >  ??              INVALID DECIMAL
000006  01 A-N-B                           >  ????            INVALID DECIMAL
        ** END **
------  ------------------------------------------------ Abend at TRITST:18 <>
000015 B  PROCEDURE DIVISION   USING  TST-REC
000016                                TYPE-OF-TRIANGLE.
000017    VALIDATE-TRIANGLE.
=====>        ADD A B GIVING A-N-B.
000019        ADD A C GIVING A-N-C.
000020        ADD B C GIVING B-N-C.
000021        IF (B-N-C NOT > A) OR (A-N-C NOT > B) OR (A-N-B NOT > C)
000022           MOVE 4 TO TYPE-OF-TRIANGLE.
```

Figure 6-1. XPEDITER/TSO Responding to a S0C7 Abend

Note: When an abend occurs in a program that **has not** been compiled with the CSS COBOL language processor (no source listing is available), the log is automatically displayed with a message indicating the action to take. For example:

```
ABEND IN MODULE WITHOUT SOURCE USE AA SNAP COMMAND FOR MORE INFORMATION
```

To assist you in determining the cause of the abend and how to correct the problem, XPEDITER/TSO provides the following:

- If Abend-AID release 7.0.2 or above is installed at your site, you can enter **AA SNAP** to display an Abend-AID Snapshot report.

  Note: In order to properly view the Snapshot report, the Abend-AID product must have been installed with LANGTYP=USAUC in the GLOBAL table. Refer to the *MVS Abend-AID Reference Manual* for additional information.

- Use the LOG command to display the diagnostic summary contained in the log.
- Use the HELP command to display help information about the abend.
- Use debugging commands to investigate the cause of the abend.

Displaying the Abend-AID Snapshot Report

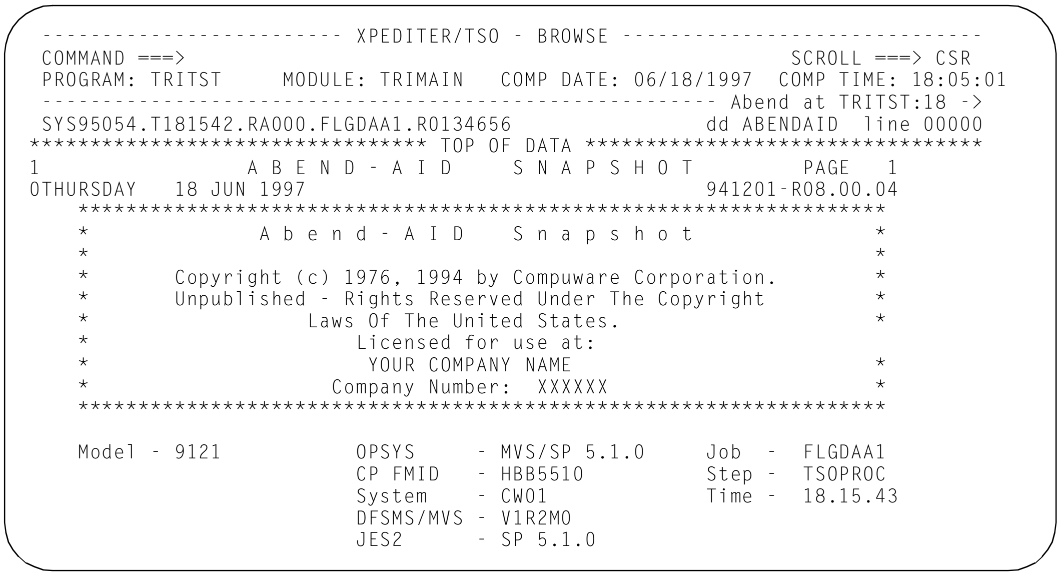
When an abend occurs in your program during an XPEDITER/TSO test session and Abend-AID release 7.0.2 or above is installed at your site, you can display an Abend-AID Snapshot report. This report contains context sensitive diagnostic information. (The report will not contain source support.)

In a nonabending situation, you can request the Snapshot report to display environment specific run-time characteristics and subsystem-related debugging information for VSAM, IDMS, IMS, and DB2 (if you have these Abend-AID options). Refer to Chapter 5, "Displaying Environmental Data" on page 5-49 for more information.

To display the Snapshot report, enter **AA SNAP** on the command line of any run-time screen. The Snapshot report is formatted according to your terminal size (80 or 133) and written to a temporary dataset DD called ABENDAID.

Note: If you want to save the abend information for printing, you must allocate the ABENDAID DD to a permanent dataset. This dataset must be RECFM=VBA and LRECL=125.

Figure 6-2 shows the header page of the Snapshot report:



```
 ------------------------ XPEDITER/TSO - BROWSE ----------------------------
 COMMAND ===>                                                SCROLL ===> CSR
 PROGRAM: TRITST     MODULE: TRIMAIN   COMP DATE: 06/18/1997  COMP TIME: 18:05:01
 ---------------------------------------------------- Abend at TRITST:18 ->
 SYS95054.T181542.RA000.FLGDAA1.R0134656              dd ABENDAID  line 00000
********************************* TOP OF DATA ********************************
1                 A B E N D - A I D     S N A P S H O T         PAGE   1
0THURSDAY   18 JUN 1997                                 941201-R08.00.04
    *******************************************************************
    *             A b e n d - A I D    S n a p s h o t                 *
    *                                                                  *
    *       Copyright (c) 1976, 1994 by Compuware Corporation.         *
    *       Unpublished - Rights Reserved Under The Copyright          *
    *                  Laws Of The United States.                      *
    *                      Licensed for use at:
    *                       YOUR COMPANY NAME                          *
    *                    Company Number:  XXXXXX                       *
    *******************************************************************

    Model - 9121           OPSYS     - MVS/SP 5.1.0     Job  -  FLGDAA1
                           CP FMID   - HBB5510          Step -  TSOPROC
                           System    - CW01             Time -  18.15.43
                           DFSMS/MVS - V1R2M0
                           JES2      - SP 5.1.0
```

Figure 6-2. Abend-AID Snapshot Report Header

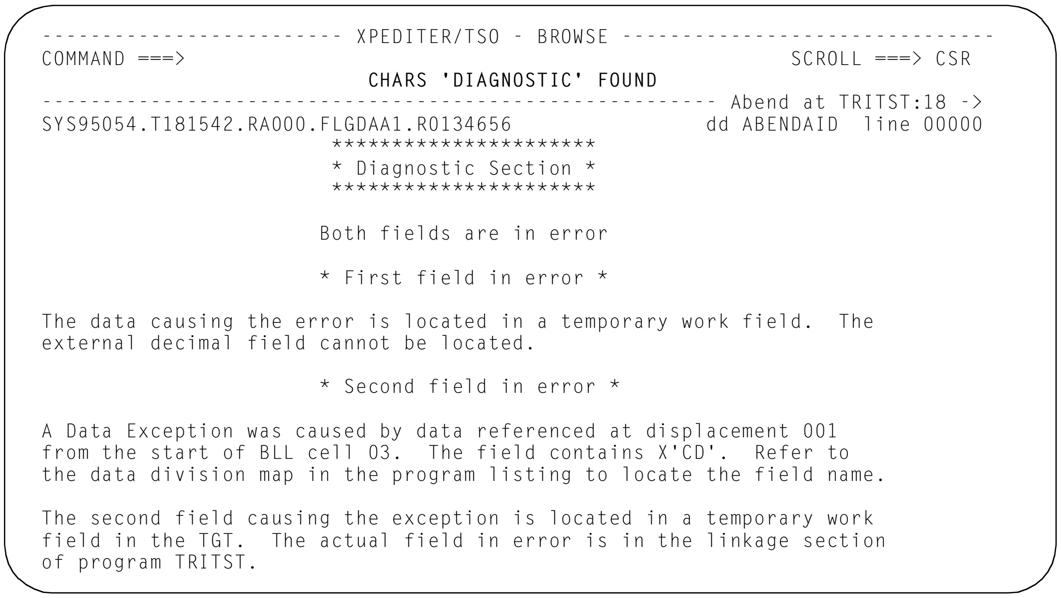
Optionally, you can display a column template at the top of the report by using SET COLS ON. The column template can be useful when viewing records or data. Enter SET COLS OFF (the default) to turn off the column template.

Browsing the Snapshot Report

The Snapshot report can be browsed using the PF8 (DOWN) and PF7 (UP) keys. You can also bypass the header page and go directly to a specific section of the Snapshot report from the source display. For example, to go directly to the Diagnostic Section, you can enter the following:

```
AA SNAP; FIND 'DIAGNOSTIC'
```

The Diagnostic Section of the report is displayed, as shown in Figure 6-3 on page 6-3.



```
------------------------ XPEDITER/TSO - BROWSE ------------------------------
COMMAND ===>                                                 SCROLL ===> CSR
                          CHARS 'DIAGNOSTIC' FOUND
---------------------------------------------------------- Abend at TRITST:18 ->
SYS95054.T181542.RA000.FLGDAA1.R0134656                dd ABENDAID  line 00000
                         **********************
                         * Diagnostic Section *
                         **********************

                        Both fields are in error

                        * First field in error *

The data causing the error is located in a temporary work field.  The
external decimal field cannot be located.

                        * Second field in error *

A Data Exception was caused by data referenced at displacement 001
from the start of BLL cell 03.  The field contains X'CD'.  Refer to
the data division map in the program listing to locate the field name.

The second field causing the exception is located in a temporary work
field in the TGT.  The actual field in error is in the linkage section
of program TRITST.
```

Figure 6-3. Snapshot Report Diagnostic Section

When you have finished viewing the Snapshot report, press **PF3** (END) to return to the source display.

If you inadvertently press **PF3** (END) while viewing the Snapshot report, you can enter **BROWSE ABENDAID** on the command line of the source or log display to redisplay the Snapshot report.

Viewing the Log

XPEDITER/TSO records the cause of the abend, the exact location, and a diagnostic summary in the log. To look at the log, type **LOG** on the command line of the Source display screen.

Note: If you have Abend-AID installed at your site, you can also display the log from the Snapshot report. An example of the diagnostic summary in the log is shown in Figure 6-4 on page 6-4.

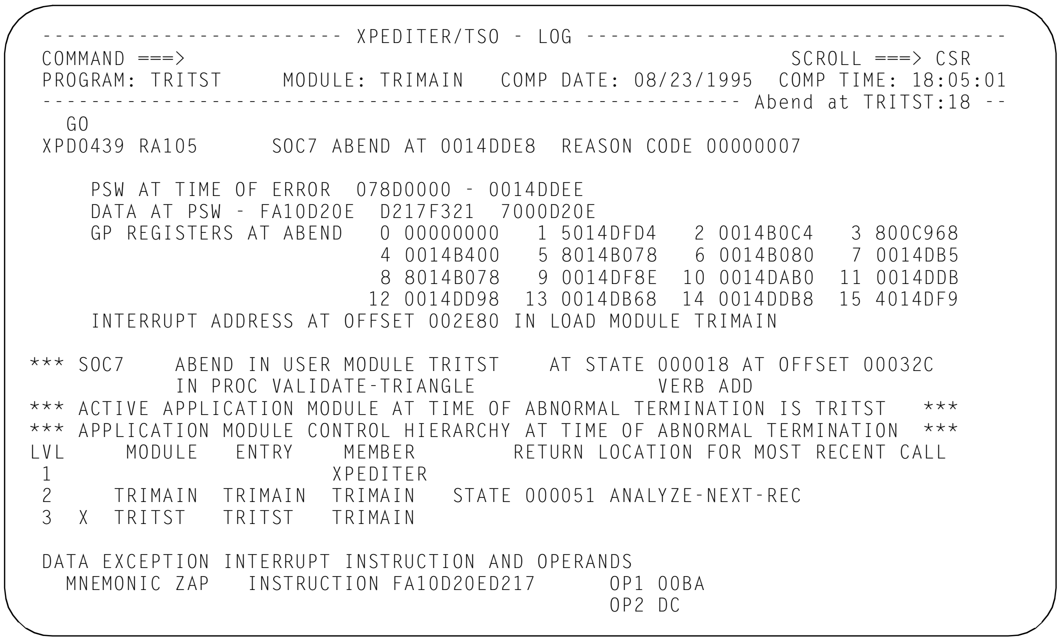

```
------------------------- XPEDITER/TSO - LOG ----------------------------------
COMMAND ===>                                              SCROLL ===> CSR
PROGRAM: TRITST     MODULE: TRIMAIN   COMP DATE: 08/23/1995  COMP TIME: 18:05:01
------------------------------------------------------------- Abend at TRITST:18 --
  GO
XPD0439 RA105      SOC7 ABEND AT 0014DDE8  REASON CODE 00000007

    PSW AT TIME OF ERROR  078D0000 - 0014DDEE
    DATA AT PSW - FA10D20E  D217F321  7000D20E
    GP REGISTERS AT ABEND   0 00000000   1 5014DFD4   2 0014B0C4   3 800C968
                            4 0014B400   5 8014B078   6 0014B080   7 0014DB5
                            8 8014B078   9 0014DF8E  10 0014DAB0  11 0014DDB
                           12 0014DD98  13 0014DB68  14 0014DDB8  15 4014DF9
    INTERRUPT ADDRESS AT OFFSET 002E80 IN LOAD MODULE TRIMAIN

*** SOC7    ABEND IN USER MODULE TRITST    AT STATE 000018 AT OFFSET 00032C
            IN PROC VALIDATE-TRIANGLE               VERB ADD
*** ACTIVE APPLICATION MODULE AT TIME OF ABNORMAL TERMINATION IS TRITST   ***
*** APPLICATION MODULE CONTROL HIERARCHY AT TIME OF ABNORMAL TERMINATION  ***
LVL     MODULE   ENTRY    MEMBER        RETURN LOCATION FOR MOST RECENT CALL
 1                        XPEDITER
 2     TRIMAIN  TRIMAIN  TRIMAIN   STATE 000051 ANALYZE-NEXT-REC
 3  X  TRITST   TRITST   TRIMAIN

 DATA EXCEPTION INTERRUPT INSTRUCTION AND OPERANDS
   MNEMONIC ZAP   INSTRUCTION FA10D20ED217      OP1 00BA
                                                OP2 DC
```

Figure 6-4. Diagnostic Summary in the Log

The first part of the diagnostic summary identifies the error message, PSW, data at PSW, general-purpose registers, offset, and the module in which the error occurred.

The middle part shows the abend type, module name, statement number, offset, paragraph name, and the failing verb. This information is followed by the module stack that identifies the last call that took place.

The last part defines the problem and gives the next sequential instruction to be executed in the Assembler format.

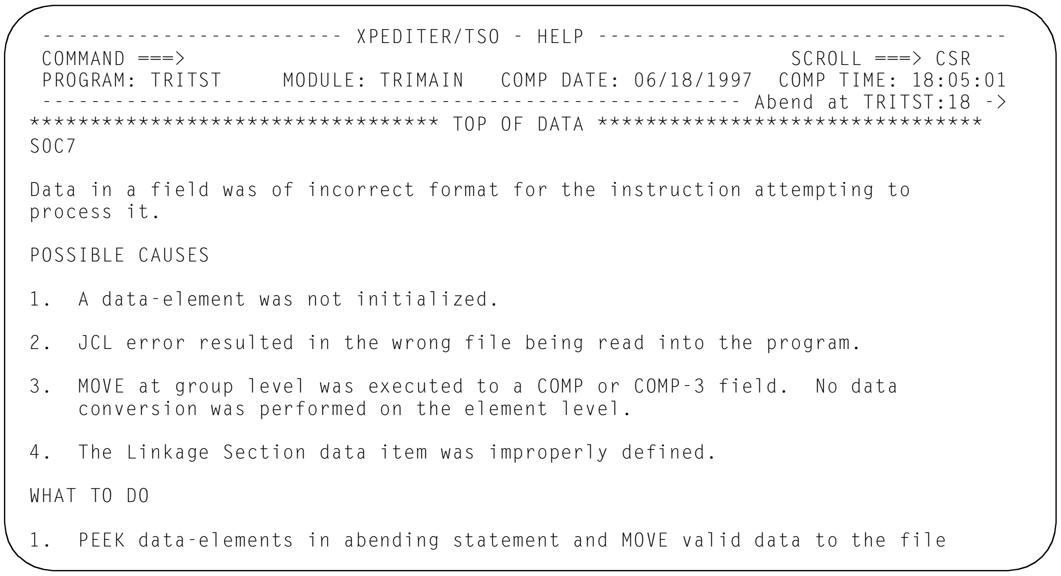
If desired, the load modules can also be displayed by entering **SHOW MODULES**.

Displaying HELP Information

The HELP command can be used to obtain further information to determine the cause and possible resolution of the abend. Any of the most commonly occurring abends (listed below) can be entered as a keyword with the HELP command.

SB14	S0C1	S0C7	S031	S806
SB37	S0C4	S001	S106	
SD37	S0C5	S013	S213	

Figure 6-5 on page 6-5 shows HELP information for a S0C7.



```
 ------------------------- XPEDITER/TSO - HELP ---------------------------------
 COMMAND ===>                                                   SCROLL ===> CSR
 PROGRAM: TRITST     MODULE: TRIMAIN   COMP DATE: 06/18/1997  COMP TIME: 18:05:01
 --------------------------------------------------------- Abend at TRITST:18 ->
********************************** TOP OF DATA ********************************
S0C7

Data in a field was of incorrect format for the instruction attempting to
process it.

POSSIBLE CAUSES

1.  A data-element was not initialized.

2.  JCL error resulted in the wrong file being read into the program.

3.  MOVE at group level was executed to a COMP or COMP-3 field.  No data
    conversion was performed on the element level.

4.  The Linkage Section data item was improperly defined.

WHAT TO DO

1.  PEEK data-elements in abending statement and MOVE valid data to the file
```

Figure 6-5. HELP Information on S0C7

Analyzing the Problem

When an abend occurs, you can use XPEDITER/TSO debugging commands to investigate the cause of the abend. For example, you can view the contents of the data referenced on the statement causing the abend, analyze the data flow and control structure, and optionally, review the path and data values that lead to the error.

Applying Fixes

Even when XPEDITER/TSO intercepts an abend, you have the opportunity to temporarily apply fixes to the problem and resume execution as if the abend had never occurred. The GO (PF12) command resumes execution. However, there may be times that you will not be able to continue normal program execution until the abend is resolved.

Using all of the information provided by the Abend-AID Snapshot report, you can experiment with possible solutions and test the fixes.

Obtaining a Memory Dump

If a dump is necessary, the SET DUMP ON command can be issued immediately to direct XPEDITER/TSO to allow the dump to be written if one of the standard dump DD files (SYSUDUMP, SYSMDUMP, or SYSABEND) is allocated to the test session. The dump file can be dynamically allocated just prior to the issuance of the SET DUMP ON command or could have been preallocated through various methods such as the File Allocation Utility (FAU), the TSO ALLOCATE command, JCL if in Batch Connect, etc. (Refer to Appendix A, "Using the File Allocation Utility" on page A-1.)

Chapter 7. Debugging With XPEDITER/IMS

XPEDITER/IMS, an IMS/DC testing and debugging product, executes Message Processing, Fast Path, and Batch Message Processing programs in an IMS-dependent region within your TSO address space. Operation of the product requires one logical TSO terminal and one logical IMS terminal (can be an ATM terminal), both on the same CPU. XPEDITER/IMS does not work with two IMS terminals.

XPEDITER/IMS is executed on Releases 3.1, 4.1, 5.1, 6.1, and 7.1 of IMS/VS.

XPEDITER/IMS does not support multiple IMS control regions that communicate with one another. However, the product can be installed on each IMS system.

Starting XPEDITER/IMS

XPEDITER/IMS consists of menus and screens accessed through ISPF. XPEDITER/IMS screens are viewed on the TSO terminal; the IMS application format screens are viewed on the IMS terminal.

Before starting the session, prepare your programs (precompile, compile, and link edit) with the Compuware Shared Services COBOL language processor. You can use the XPEDITER/TSO online facilities to prepare your programs. Refer to Chapter 4, "Preparing Your Programs" on page 4-3 for additional information. For more information about Compuware's Shared Services, refer to the *Compuware Shared Services Installation and Customization Guide*.

1. Once you have a valid load module and DDIO dataset, log on to TSO and invoke XPEDITER/TSO. Use the procedures that are in effect at your site.
2. From the XPEDITER/TSO Primary Menu, select option **2** (TSO) to display the Environments Menu.

   Note: If the Environments Menu is not displayed, access it by entering **SETUP** on the command line of the test screen that is being displayed. On the Test Setup Menu, enter option **0** (ENVIRONMENT).
3. Specify the appropriate environment, **8** (MPP) or **9** (BMP/IFP) on the Environments Menu. The environment test screen for that environment is displayed.
4. On the environment test screen, enter the command **SETUP** to display the Setup Menu.
5. Specify **A** (All) to review all your setup selections.
   - Load libraries—user program libraries allocated as STEPLIB. Verify that the order of concatenation is correct.
   - DDIO dataset—library name should be the dataset name specified on the CWPDDIO DD statement in your XPEDITER compile step.
   - Test script libraries.
   - Test session log dataset.
   - Test session script dataset.
   - DB2 system names and DSNLOAD datasets.
   - PANEXEC defaults (if installed).

- PSB and DBD libraries (GSAM only).
- IMS preload list.
- IMS authorized load libraries.
- IMS region ID and PARM strings—PARM fields set up at install time can be changed.

6. If everything is correct, press **PF3** from the Setup Menu to return to the environment test screen.
7. Complete the environment test screen as described in “Debugging an MPP Program” or “Debugging a BMP/IFP Program” on page 7-6.

   Note: XPEDITER/IMS environment test screens shown as examples in this section are fully described in Appendix B, "XPEDITER/TSO Environment Test Screens" on page B-1.
8. Wait for the intercepts to be set, then log on to IMS and enter the transaction(s).
9. View the XPEDITER/IMS source display on the TSO terminal. Switch to the IMS terminal for all input and output operations.
10. At the end of each MPP transaction, enter **GO** to continue debugging or **EXIT** to terminate.
11. At the end of a Batch Message Processing (BMP) or Interactive Fast Path (IFP) program, you are returned to the BMP/IFP screen. Press **Enter** to go into the same debugging session again, or enter **END** to return to the Primary Menu.

Debugging an MPP Program

Select option **8** from the Environments Menu to debug an MPP program. The MPP Program environment test screen shown in Figure 7-1 is displayed. The MPP screen lets you set up environment parameters for debugging a program in an IMS Message Region. When you identify the transactions to be debugged and initiate the session, XPEDITER/IMS attaches the IMS message region within the TSO address space.

Enter **END** to terminate any display. HELP is also available from any screen.

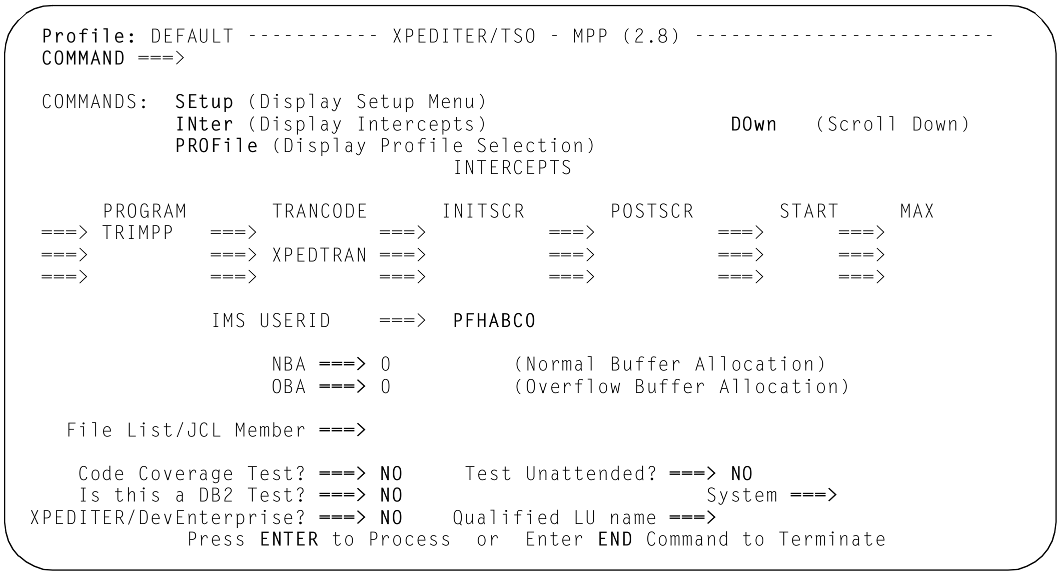

```
 Profile: DEFAULT ----------- XPEDITER/TSO - MPP (2.8) -------------------------
 COMMAND ===>

 COMMANDS:  SEtup (Display Setup Menu)
            INter (Display Intercepts)                  DOwn   (Scroll Down)
            PROFile (Display Profile Selection)
                                   INTERCEPTS

      PROGRAM       TRANCODE      INITSCR       POSTSCR       START     MAX
 ===> TRIMPP   ===>          ===>          ===>          ===>      ===>
 ===>          ===> XPEDTRAN ===>          ===>          ===>      ===>
 ===>          ===>          ===>          ===>          ===>      ===>

               IMS USERID    ===>  PFHABCO

                    NBA ===> 0          (Normal Buffer Allocation)
                    OBA ===> 0          (Overflow Buffer Allocation)

   File List/JCL Member ===>

    Code Coverage Test? ===> NO     Test Unattended? ===> NO
    Is this a DB2 Test? ===> NO                        System ===>
XPEDITER/DevEnterprise? ===> NO    Qualified LU name ===>
             Press ENTER to Process  or  Enter END Command to Terminate
```

Figure 7-1. MPP Program Test Screen

The following commands are valid on the MPP Program test screen:

SEtup
Displays the Setup Menu from which you can select screens for specifying load libraries, DDIO datasets, and so on.

Log
Browses the session log *after* you execute a program. The log file contains a record of the commands that were entered during the debugging session and the responses to them.

INter
Displays the Intercepts screen showing you what IDs are logged on to XPEDITER/IMS, what programs are being used, and so on.

DOwn
Scrolls to a continuation MPP screen. From this screen, UP is available for returning to the first debugging screen.

PROFile
Displays the available Profile options in a file where you can select, delete, copy, rename, and Use the profile of your choice. It also lets you "merge" or "copy" another user(s) profile into your own.

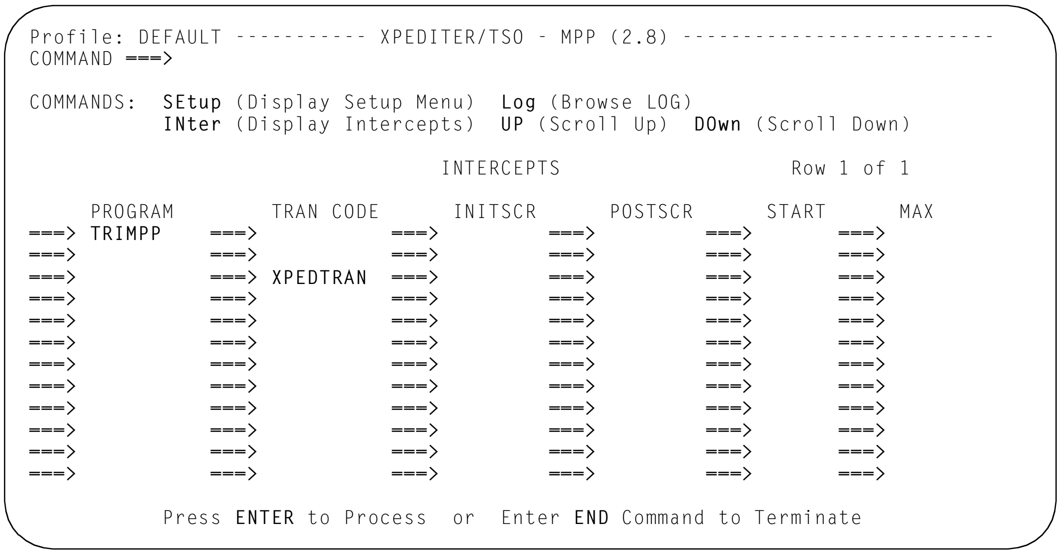
ALLOC
This command takes you to the File Allocation Utility (FAU) screens to create a file allocation list, which your program may require for the debugging session.

The fields on the MPP Program test screen are:

PROGRAM
Required if TRAN CODE is not specified. Used to identify which programs are to be debugged by XPEDITER/IMS. You can enter the load module name in this field or the transaction code in the TRAN CODE field, or supply values for both fields. If you enter only the program name, the program **must** be associated with only one transaction. If the program is associated with multiple transactions, then the transactions that are to be debugged must be entered in the TRAN CODE field and the program name can be omitted from the PROGRAM field.

Three programs or transaction codes can be specified on the first MPP screen. If you fill the first screen and need more space, enter **DOWN** (PF8), and a second (continuation) screen is displayed. Refer to Figure 7-2.

Once the second screen is full, you can scroll down to a third screen. The third screen is not displayed in this chapter. Notice that on the second (and subsequent screens), the UP command is available in addition to the DOWN command.



```
Profile: DEFAULT ---------- XPEDITER/TSO - MPP (2.8) -------------------------
COMMAND ===>

COMMANDS:  SEtup (Display Setup Menu)  Log (Browse LOG)
           INter (Display Intercepts)  UP (Scroll Up)  DOwn (Scroll Down)

                                  INTERCEPTS                   Row 1 of 1

     PROGRAM        TRAN CODE       INITSCR       POSTSCR       START      MAX
===> TRIMPP    ===>            ===>          ===>          ===>        ===>
===>           ===>            ===>          ===>          ===>        ===>
===>           ===> XPEDTRAN   ===>          ===>          ===>        ===>
===>           ===>            ===>          ===>          ===>        ===>
===>           ===>            ===>          ===>          ===>        ===>
===>           ===>            ===>          ===>          ===>        ===>
===>           ===>            ===>          ===>          ===>        ===>
===>           ===>            ===>          ===>          ===>        ===>
===>           ===>            ===>          ===>          ===>        ===>
===>           ===>            ===>          ===>          ===>        ===>
===>           ===>            ===>          ===>          ===>        ===>
===>           ===>            ===>          ===>          ===>        ===>

           Press ENTER to Process  or  Enter END Command to Terminate
```

Figure 7-2. Second MPP Program Test Screen

TRAN CODE
Used to identify which transactions are to be debugged by XPEDITER/IMS. You can enter the transaction code in this field or the load module name in the PROGRAM field, or enter values for both fields.

This field is required if PROGRAM is not specified or if multiple transactions are associated with the program.

INITSCR
The member name of the script in the INCLUDE library specified on the Setup panel. The INITSCR field can be typed over to specify a test script member which can be processed at the beginning of a debugging session.

When using XPEDITER/IMS, enter the member name of a test script if you want to execute a predefined command stream at the *beginning* of the debugging session. This member will be executed after the inclusion of the Site-wide script member @@SITE@@, if defined.

POSTSCR
Enter the member name of a test script if you want to execute a predefined command stream at the *end* of the debugging session.

START
Enter up to four digits to specify the occurrence of the program invocation on which the intercept is to begin. By default, program intercept begins on the first occurrence. When the program is intercepted, the debugging session is initiated and the source is displayed.

MAX
Enter up to four digits to specify the maximum number of times the program intercept is to be processed. If this field is left blank, the value defaults to an infinite number. If you enter an EXIT command and there are still intercepts remaining, those intercepts are ignored.

IMS USERID
Enter the userid of the XPEDITER/IMS client initiating the test or debugging session, up to ten digits, provided XPEDITER/Code Coverage has been installed at your site.

NBA
Enter the Normal Buffer Allocation (NBA), up to two digits. The default value is 0.

OBA
Enter the Overflow Buffer Allocation (OBA), up to two digits. The default value is 0.

Note: Values are inserted in the PARM string when the IMS region is attached.

File List/JCL Member
Enter the dataset name that contains the file list, CLIST, or JCL. The File Allocation Utility (FAU) preallocates files and databases that will be processed by the program upon entry to the debugging session.

If the member name of a PDS is omitted, a member list is displayed.

If the dataset contains a CLIST, XPEDITER/IMS immediately executes it and begins the debugging session. If the dataset contains a file list or JCL, the FAU is invoked to dynamically allocate the specified files before beginning the debugging session. Refer to Appendix A, "Using the File Allocation Utility" on page A-1 for detailed information about the FAU.

Note: If your site does not use OS/VS files for input or output, you do not have to make an entry in the File List/JCL Member field.

Code Coverage Test?
Enter **YES** if XPEDITER/Code Coverage data should be collected for this test. The default value is NO.

Note: The Code Coverage option is only available if you have purchased XPEDITER/Code Coverage.

Test Unattended?
Enter **YES** in the 'Test Unattended?' field to run in Unattended mode. The default value is NO. In unattended mode, after the XPEDITER/IMS Message Processing Region (MPR) is attached, only XPEDITER commands are processed in the Initial Script, Post Script, and Abend Script. The TSO terminal remains locked and XPEDITER commands are not allowed from the terminal. When you are finished testing in the XPEDITER/IMS MPR, the MPR may be stopped and the TSO terminal unlocked by using either the XPEDITER Stop Region transaction, XPST, or the XPEDITER Stop Region BMP procedure XPSTOP. Refer to "Stopping the XPEDITER/IMS Dependent Region" on page 7-11.

Is This a DB2 Test?
Enter **YES** if the program executes EXEC SQL statements. The default value is NO.

System
Enter the DB2 subsystem name if the program executes EXEC SQL statements. The subsystem name depends on the release level of DB2 and is assigned at the time of installation.

XPEDITER/DevEnterprise?
If you want to connect to XPEDITER/DevEnterprise, enter YES in this field. The default is NO.

Qualified LU name
If you are going to use XPEDITER/DevEnterprise, enter the fully qualified LU name. This should be the same name that was entered on the installation screen at install time.

When entries for the screen are complete and you press **Enter**, the screen clears. If the intercepts are set, the following messages appear on the screen:

```
THE IMS INTERCEPTS ARE BEING SET
INTERCEPTS SET - STARTING THE IMS REGION
THE TEST TRANSACTION CAN BE ENTERED
```

The terminal is then locked. If an error is encountered, however, the intercepts are not set and you are returned to the test screen. A message indicating an error occurred is displayed on the message line. You can access the log for more information.

Note: There are several reasons the intercepts might not be set; for example:

- You may have entered a program type other than TP (Message Processing Program).
- The program may be TP, but someone else is debugging the same transaction. In a message processing region, two users cannot debug the same transaction at the same time. XPEDITER/IMS changes the class code of each transaction, but it does not change the high-level program associated with the transaction. Therefore, if the transaction is being used by another person, you receive an error message.

If the intercepts are set successfully, go to the IMS terminal and start the transaction you want to debug by entering the transaction code that invokes your program. Note that you should invoke the program the way you would without XPEDITER/IMS; e.g., by going through a signon screen or another transaction. If a format screen is invoked, enter any data needed by the transaction.

When you press **Enter**, IMS schedules the transaction for execution and locks your terminal if it is in response mode. Return to the TSO terminal.

If you have a source listing member for the high-level program, your source will be displayed. Refer to Chapter 5, "Debugging Interactively" on page 5-1 for information on how to interactively debug a program with source. All interactive XPEDITER/TSO commands are valid except USE and RETEST. If you do not have a source listing member for the high-level program, the log is displayed with a message indicating that the program does not have a source listing member. Refer to Chapter 5, "Debugging a Sourceless Program" on page 5-46 for information on how to interactively debug a program without a source listing member.

To enter or review data at any point in execution, go to the IMS terminal.

At the `TEST COMPLETED` message, you can enter **GO** to retest this transaction or test another one, or enter **EXIT** to return to the MPP screen. At the end of the debugging session, the log is available for viewing. You can also save a script of the commands entered for future use.

Debugging a BMP/IFP Program

After selecting option **9** from the Environments Menu to debug a BMP or Fast Path program, the BMP/IFP Program screen shown in Figure 7-3 is displayed. The BMP/IFP screen lets you set up the parameters to test and debug a program in an IMS, BMP, or Fast Path Region. When you identify the transactions to be debugged and initiate a session, XPEDITER/IMS attaches the IMS message region within the TSO address space.

The commands (except DOWN) described in “Debugging an MPP Program” on page 7-2 can also be entered on this screen.

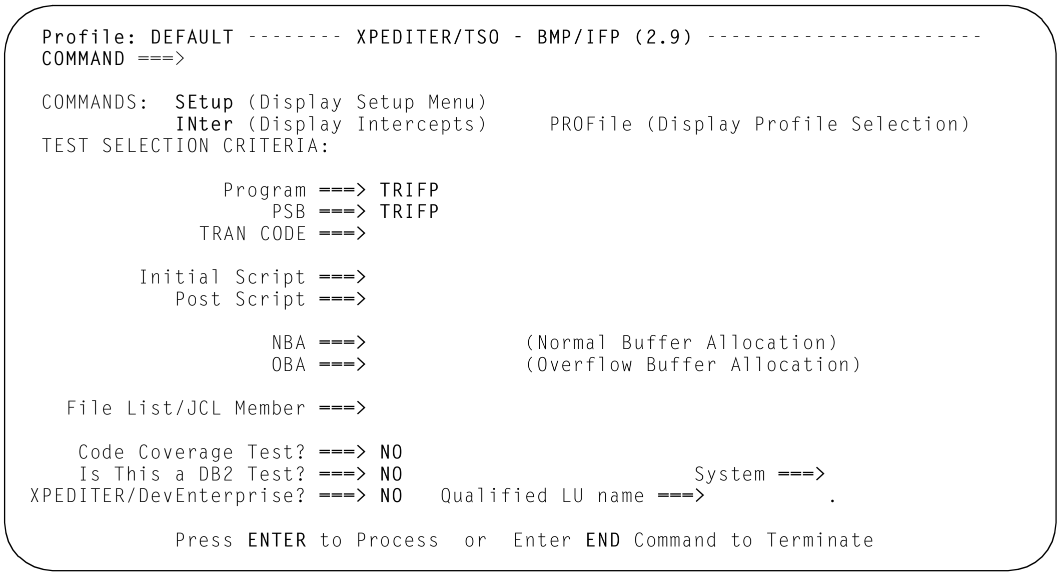

```
 Profile: DEFAULT -------- XPEDITER/TSO - BMP/IFP (2.9) ----------------------
 COMMAND ===>

 COMMANDS:  SEtup (Display Setup Menu)
            INter (Display Intercepts)     PROFile (Display Profile Selection)
 TEST SELECTION CRITERIA:

                Program ===> TRIFP
                    PSB ===> TRIFP
              TRAN CODE ===>

         Initial Script ===>
            Post Script ===>

                    NBA ===>             (Normal Buffer Allocation)
                    OBA ===>             (Overflow Buffer Allocation)

   File List/JCL Member ===>

    Code Coverage Test? ===> NO
    Is This a DB2 Test? ===> NO                          System ===>
XPEDITER/DevEnterprise? ===> NO   Qualified LU name ===>          .

            Press ENTER to Process  or  Enter END Command to Terminate
```

Figure 7-3. BMP/IFP Program Screen

The fields on the BMP/IFP screen are:

Program
Required. Used to identify which program is to be intercepted and debugged by XPEDITER/IMS. You must enter the load module name in this field or supply values for both this field and the TRAN CODE field. XPEDITER inserts this value into the program subparameter for the parameter it passes to IMS.

PSB
Required. Enter the program specification block associated with the program you are debugging. XPEDITER inserts the specified value into the BMP parameter passed to the IMS driver program.

TRAN CODE
Used with a BMP to provide the name of a message queue to be processed by the BMP. This field is not used with an IFP program.

Initial Script
The member name of the script in the INCLUDE library specified on the Setup panel. The INITSCR field can be typed over to specify a test script member which can be processed at the beginning of a debugging session.

Enter the member name of a test script if you want to execute a predefined command stream at the *beginning* of the debugging session. This member will be executed after the inclusion of the Site-wide script member @@SITE@@, if defined.

Post Script
Enter the member name of a test script if you want to execute a predefined command stream at the *end* of the debugging session.

NBA
Enter the Normal Buffer Allocation (NBA), up to two digits.

OBA
Enter the Overflow Buffer Allocation (OBA), up to two digits.

Note: The values are inserted in the PARM string when the IMS region is attached.

File List/JCL Member
Enter the dataset name that contains the file list, CLIST, or JCL. The File Allocation Utility (FAU) preallocates files and databases that will be processed by the program upon entry to the debugging session.

If the member name of a PDS is omitted, a member list is displayed. If the dataset contains a CLIST, XPEDITER/IMS immediately executes it and begins the debugging session. If the dataset contains a file list or JCL, the FAU is invoked to dynamically allocate the specified files before beginning the debugging session. Refer to Appendix A, "Using the File Allocation Utility" on page A-1 for detailed information about the FAU.

Note: If your site does not use OS/VS files for input or output or GSAM databases, do not make an entry in the File List/JCL Member field.

Code Coverage Test?
Enter **YES** if XPEDITER/Code Coverage data should be collected for this test. The default value is NO.

Note: The Code Coverage option is only available if you have purchased XPEDITER/Code Coverage.

Is This a DB2 Test?
Enter **YES** if the program executes EXEC SQL statements. The default value is NO.

System
Enter the DB2 subsystem name if the program executes EXEC SQL statements. The subsystem name depends on the release level of DB2 and is assigned at the time of installation.

XPEDITER/DevEnterprise?
If you want to connect to XPEDITER/DevEnterprise, enter YES in this field. The default is NO.

Qualified LU name
If you are going to use XPEDITER/DevEnterprise, enter the fully qualified LU name. This should be the same name that was entered on the installation screen at install time.

Press **Enter** to initiate the debugging session.

When all values have been entered and you press **Enter**, the screen clears. If the intercepts are set, the following messages appear on the screen:

```
THE IMS INTERCEPTS ARE BEING SET
INTERCEPTS SET - STARTING THE IMS REGION
```

The terminal is then locked until the screen changes. If an error is encountered, you are returned to the test screen. A message indicating an error occurred is displayed on the message line. You can access the log for more information.

When the intercepts are set, go to the IMS terminal and begin the transaction you want to debug by entering the transaction code that invokes your program. Note that you should invoke the program in the usual way (without XPEDITER/IMS); e.g., going through a sign-on screen or other transaction if necessary. Once the format screen appears, you can enter any data needed by the transaction. IMS schedules the transaction for execution, then locks the IMS terminal if it is in response mode. Return to the TSO terminal.

Note: If you are debugging a non-Wait-For-Input message-driven BMP, the transaction for the program must be entered before the first GU to the message queue is executed.

If you have a source listing member for the high-level program, your source will be displayed. Refer to Chapter 5, "Debugging Interactively" on page 5-1 for additional information on how to interactively debug a program with source. All interactive XPEDITER/TSO commands are valid except USE and RETEST.

If you do not have a source listing member for the high-level program, the log is displayed with a message indicating that the program does not have a source listing member. Refer to "Debugging a Sourceless Program" on page 5-46 for information on how to interactively debug a program without a source listing member.

To enter or review data at any point in execution, go to the IMS terminal.

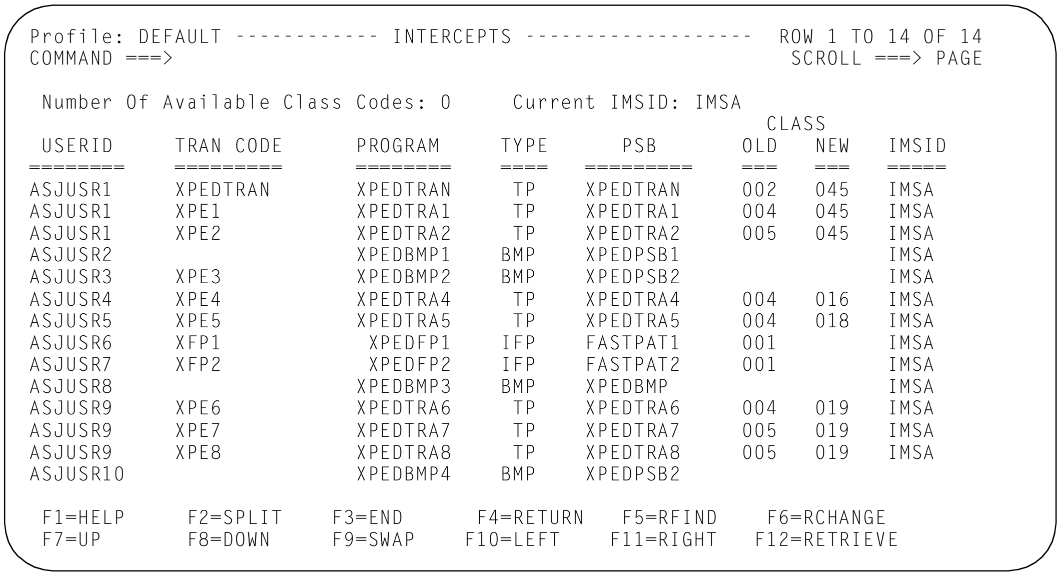
To end a BMP debugging session, type **EXIT**. To end an Interactive Fast Path (IFP) debugging session, the directions are different. You can type **EXIT** if you have a breakpoint. If the Interactive Fast Path (IFP) program is waiting for input from the terminal, the IFP region may be stopped and the TSO terminal unlocked by using either the XPEDITER Stop Region transaction, XPST, or the XPEDITER Stop Region BMP procedure XPSTOP. Refer to “Stopping the XPEDITER/IMS Dependent Region” on page 7-11.

When the debugging session is terminated, you are returned to the BMP/IFP screen. At this point, log and script datasets are available. You can press **Enter** to go into the same debugging session again, or type **END** to go back to the Environments Menu.

What to Do When Intercepts Cannot Be Set

When intercepts cannot be set and the message `MAX USERS` is displayed on the test screen, enter **INTER** on the command line to look at the Intercepts screen, shown in Figure 7-4. The Number Of Available Class Codes field is zero (0), indicating that the maximum number of users are testing with XPEDITER/IMS.

The Intercepts screen provides valuable information regarding the use of XPEDITER/IMS—how many users can still test using XPEDITER/IMS, what programs/transactions are being tested, and other information connected with the program and the transaction code.



```
Profile: DEFAULT ----------- INTERCEPTS ------------------  ROW 1 TO 14 OF 14
COMMAND ===>                                                 SCROLL ===> PAGE

 Number Of Available Class Codes: 0     Current IMSID: IMSA
                                                           CLASS
 USERID     TRAN CODE      PROGRAM     TYPE      PSB      OLD   NEW   IMSID
========    =========      ========    ====   =========   ===   ===   =====
ASJUSR1     XPEDTRAN       XPEDTRAN     TP    XPEDTRAN    002   045   IMSA
ASJUSR1     XPE1           XPEDTRA1     TP    XPEDTRA1    004   045   IMSA
ASJUSR1     XPE2           XPEDTRA2     TP    XPEDTRA2    005   045   IMSA
ASJUSR2                    XPEDBMP1    BMP    XPEDPSB1                IMSA
ASJUSR3     XPE3           XPEDBMP2    BMP    XPEDPSB2                IMSA
ASJUSR4     XPE4           XPEDTRA4     TP    XPEDTRA4    004   016   IMSA
ASJUSR5     XPE5           XPEDTRA5     TP    XPEDTRA5    004   018   IMSA
ASJUSR6     XFP1            XPEDFP1    IFP    FASTPAT1    001         IMSA
ASJUSR7     XFP2            XPEDFP2    IFP    FASTPAT2    001         IMSA
ASJUSR8                    XPEDBMP3    BMP    XPEDBMP                 IMSA
ASJUSR9     XPE6           XPEDTRA6     TP    XPEDTRA6    004   019   IMSA
ASJUSR9     XPE7           XPEDTRA7     TP    XPEDTRA7    005   019   IMSA
ASJUSR9     XPE8           XPEDTRA8     TP    XPEDTRA8    005   019   IMSA
ASJUSR10                   XPEDBMP4    BMP    XPEDPSB2

 F1=HELP     F2=SPLIT    F3=END      F4=RETURN   F5=RFIND    F6=RCHANGE
 F7=UP       F8=DOWN     F9=SWAP    F10=LEFT    F11=RIGHT   F12=RETRIEVE
```

Figure 7-4. Intercepts Screen

The Intercepts screen does not contain any input fields. All of the information is displayed for you. The fields on the screen are:

Number Of Available Class Codes
The number of class codes available. This field tells you how many more people can run an MPP test. Notice that Figure 7-4 on page 7-8 displays the number of class codes available for use as zero (0), indicating that the maximum number of MPP users are currently testing with XPEDITER/IMS.

Current IMSID
The name of the IMS control region with which you are communicating.

USERID
The user ID of the person running the test.

TRAN CODE
The transaction code associated with the program, if one was used.

PROGRAM
The name of the program being tested.

TYPE
The type of program being tested.

PSB
The PSB name associated with the program. Often, this name is the same as the program name.

OLD CLASS
The original class code of the user's transaction.

NEW CLASS
The XPEDITER/IMS reserved class code for the transaction. XPEDITER/IMS reassigns the class codes for Message Processing transactions.

IMSID
The name of the IMS control region with which the IMS user programs communicate. This name does not have to be the current IMSID.

Functions Supported

Database support is available with XPEDITER/IMS through the IMS control region. All databases are supported, along with the following types of transactions:

- Conversational and nonconversational
- Response and nonresponse mode
- Wait-for-Input and non-Wait-For-Input

The type of transaction you can process is determined by the way your IMS system is defined.

Conversational and Nonconversational Transactions

Programs that process nonconversational transactions have no knowledge of previous transactions or responses. Programs that process conversational transactions retain knowledge of each transaction in a scratch pad area. When each succeeding transaction is received, the scratch pad area is read to determine what happened during the previous execution of that program.

In XPEDITER/IMS, the two transaction types are processed the same way—there is no difference between the two.

Response Mode and Nonresponse Mode

When response mode is in effect, IMS does not accept any input from the communication line or terminal until the program has sent a response to the previous input message. The originating terminal is unusable (i.e., the keyboard locks) until a reply is sent back to the terminal.

When nonresponse mode is in effect, the keyboard is not locked. Multiple requests can be sent before a response is received.

WFI and Non-WFI Transactions

When a program is defined as processing Wait-For-Input (WFI) transactions, the program remains in main storage, even when there are no more messages for it to process. This ensures that the program and region are immediately available when a transaction comes in.

If a program is defined as processing non-Wait-For-Input (non-WFI) transactions, the program can process one or more transactions and end, then have to start again when another transaction is received for processing.

Types of Programs You Can Debug

With XPEDITER/IMS, you can debug three types of programs:

- Message Processing Programs (MPPs)
- Batch Message Processing (BMP) Programs
- Message-driven Interactive Fast Path (IFP) Programs

These three types of programs can access various resources under XPEDITER/IMS. Available to them are full function databases, DB2, GSAM (BMP only), and Fast Path databases.

Message Processing Programs

The primary purpose of MPPs is to quickly process your requests from the terminal or from another application program. Ideally, the requests are small and processing is set up to respond to them quickly. MPPs process messages as their input and send messages as responses.

An MPP can access full function databases, Fast Path databases, and DB2 databases. An MPP cannot access GSAM databases.

In XPEDITER/IMS, multiple MPP transactions can be debugged in the same session.

Batch Message Processing Programs

You might use a BMP program when processing time is long, but an immediate response is not required. There are two types of BMP programs—message-driven and nonmessage-driven. A message-driven BMP program is used to process information received from a terminal or other program, but held by IMS in a message queue.

A nonmessage-driven program is often used when no access to messages is required. It is also typically a long-running program, used when you have a large number of updates to do or when running a report.

Both types of BMP programs can access OS/VS files, GSAM databases, full function databases, Fast Path databases, and DB2 databases.

In XPEDITER/IMS, a BMP program is executed in its own region, and when the program is terminated, the region is also terminated.

Fast Path Programs

An IFP program functions the same way an MPP does, but it is scheduled by IMS in a manner similar to a BMP program. An IFP program increases the performance of certain types of database applications. It provides efficient access to large volumes of data that are broken up into areas that can be accessed independently of each other.

Note: IFP programs, like MPP and BMP programs, can access Fast Path databases, as well as full function databases and DB2 databases.

In XPEDITER/IMS, an IFP program is executed in a Fast Path region, and when the program is terminated, so is the region.

Using XPEDITER/IMS Effectively

Some information in the following areas can help you run MPP, BMP, and IFP debugging sessions effectively:

- MPP test setup
- Scheduling difficulties
- Interrupting a debugging session
- Fast Path considerations

MPP Test Setup

For an MPP debugging session, if a PSB is defined to your IMS system as capable of handling more than one transaction, you must enter the specific transaction code you want to debug.

Scheduling Difficulties

There are two XPEDITER/IMS capabilities that can help if you are unable to schedule a transaction or program:

- The Intercepts display invoked by the INTER command on your test screen
- The session log

When you cannot schedule a particular transaction, another person may be using it. The transaction a user sets up, either by entering a transaction code (MPP, BMP, or IFP) or through the PSB name (IFP) or program name (MPP), is allocated to that user for the duration of the session. The Intercepts display tells you the transactions that are allocated and the users to whom they are allocated.

XPEDITER/IMS records in the log many conditions that prevent a transaction or program from being scheduled. If, after setting intercepts, you still receive the test screen, browse the log for more information. XPEDITER/IMS can determine whether a program or transaction is locked or stopped, or not able to be scheduled for other reasons. XPEDITER/IMS can also capture IMS scheduling abends, which are noted in the log. Without XPEDITER/IMS, these conditions can cause your IMS terminal to lock or not respond, requiring master terminal operator intervention to find the cause of the problem.

Stopping the XPEDITER/IMS Dependent Region

An XPEDITER/IMS Dependent Region is categorized as: (1) an MPP region, (2) a BMP region, or (3) an IFP region. There are several different ways of stopping an XPEDITER/IMS Dependent Region. The circumstances depend on whether the dependent region is stopped at an XPEDITER breakpoint, the dependent region is in a CPU loop, or the dependent region is waiting for the next input message.

If the region is stopped at an XPEDITER breakpoint, you may enter the **XPEDITER EXIT** command to stop the XPEDITER/IMS dependent region. Before entering the **EXIT** command, you may roll back any/all uncommitted IMS database updates, DB2 table updates, and system generated messages by issuing the following command:

```
DLI ROLB IOPCB
```

If the XPEDITER/TSO terminal session is not available, the region may be stopped by using the XPEDITER/IMS Detach Region Facility. Refer to "XPEDITER/IMS Detach Region Facility" on page 7-13.

If the region is in a CPU loop, you may use the **ATTN** key to stop the XPEDITER/IMS dependent region. You may press the **ATTN** key once and wait for XPEDITER/IMS to process the attention interrupt. Attention key processing can be delayed under certain predefined conditions. For example, if the **ATTN** key is pressed while the XPEDITER/IMS dependent region is executing a DL/I database call or waiting for an input message, attention processing is deferred.

If an MPP or IFP region is waiting for the next input message, the XPEDITER/IMS dependent region may be stopped by using either the XPEDITER Stop Region transaction, trancode XPST, or the XPEDITER Stop Region BMP procedure XPSTOP.

To stop the region from an IMS terminal, enter trancode XPST. Transaction XPST displays a formatted screen. On the screen, a field is displayed for entering the Jobname or TSO-ID to be stopped. When the screen is displayed, this field will contain your IMS logon id, since many users logon to IMS with their TSO ID. If the installation allows it, you may override the default Jobname shown on the screen with a completely different Jobname of an XPEDITER MPP or IFP region. When you press **Enter** while accessing the formatted screen, the transaction will issue the following IMS Automated Operator Interface command: `/STOP REG reg#` where **reg#** is the region identifier associated with the specified Jobname.

To stop the region from a TSO terminal, submit a job which executes your site's BMP procedure to run the XPSTOP program. The XPEDITER jobname or TSO Userid of the region to be cancelled is specified on either the IMS APARM or in the first eight bytes of a control card in a file with DDNAME XPPARM. The APARM technique is usually more convenient; however, releases of IMS prior to IMS 3.1 do not support APARM.

The following JCL examples may be used to invoke a BMP job to stop the TSO Userid TSOUSR1. IMSBATCH is an IBM supplied IMS procedure that is included in the IMS PROCLIB. Your Systems programmer may have created a version of this procedure which is modified for your site. In both cases, the programs executed by XPSTOP must be made available to the region by placing the program code in a library pointed to be the STEPLIB DD statement. This can be done by copying XPSTOP, ADSIM015, and ADSRA093 from the XPEDITER/TSO LOADLIB to a user library included in STEPLIB or by adding the XPEDITER/TSO LOADLIB to the STEPLIB concatenation.

The two JCL examples are displayed as follows:

```
//  Your jobcard
//stepname EXEC IMSBATCH,MBR=XPSTOP,PSB=XPSTOP,IN=XPSTOP,
//         APARM=TSOUSR1
//procedurestepname.STEPLIB DD   (existing DD statement)
//         DD DSN=xpediter/tso.LOADLIB
//XPPRINT DD SYSOUT=*
//
         or

//  Your jobcard
//stepname EXEC IMSBATCH,MBR=XPSTOP,PSB=XPSTOP,IN=XPSTOP
//procedurestepname.STEPLIB DD   (existing DD statement)
//         DD DSN=xpediter/tso.LOADLIB
//XPPRINT DD SYSOUT=*
//XPPARM DD *
TSOUSR1
//
```

Fast Path Caution

If your IMS system includes a Fast Path exit routine to route transactions to programs not originally genned to handle these transactions, you can experience the following difficulties:

- The program being executed can receive a transaction you did not enter.
- A transaction you entered can be routed to another program in another region. In this case, no source code displayed. If the other program is active, the transaction is processed or rejected with an appropriate message.

Abends and Recovery Processing

If an abend occurs, XPEDITER/IMS provides the following benefits:

- If the application program abends, XPEDITER/IMS intercepts the abend and displays a message indicating that information about the abend can be viewed by accessing the Abend-AID Snapshot report (only if Abend-AID release 7.0.2 or above is installed) or the log.
- You can generate a memory dump in the same manner as XPEDITER/TSO. If a dump is necessary, the SET DUMP ON command can be issued immediately to direct XPEDITER/TSO to allow the dump to be written if one of the standard dump DD files (SYSUDUMP, SYSMDUMP, or SYSABEND) is allocated to the test session. The dump file can be dynamically allocated just prior to the issuance of the SET DUMP ON command or could have been preallocated through various methods such as the File Allocation Utility (FAU), the TSO ALLOCATE command, JCL if in Batch Connect, etc. (Refer to Appendix A, "Using the File Allocation Utility" on page A-1.)
- If the IMS dependent region comes down for any reason, XPEDITER/IMS intercepts the abend.

Note: When debugging a message-driven transaction, IMS generates a U468 abend if you exit the session before the first GU to the message queue. If the transaction is in response mode, the terminal node must be restarted by the appropriate operator.

If you require a dump from an XPEDITER/IMS debugging session, you must allocate the ddname XPIMSDMP to bypass the XPEDITER abend interception. Since this is a TSO allocation, it can be entered from the command line as follows:

```
TSO ALLOC FI(XPIMSDMP) DUMMY
```

It can also be allocated from within XPEDITER/IMS by issuing the ALLOC command and using the File Allocation Utility.

XPEDITER/IMS Detach Region Facility

When using XPEDITER in an IMS environment, there can be times when you need to force the end of an XPEDITER test; e.g., a programmer goes home without ending the test. In some releases of IMS, the TSO ID cannot be cancelled while the IMS MPP/BMP/IFP region is still active.

For this case and others, a batch job executing the detach utility program ADSIM012 can be run to force the end of a user's XPEDITER IMS MPP, BMP, or IFP test. If desired, the TSO ID can be dropped at this time.

Set up the batch job to execute the ADSIM012 program as shown below:

```
//XXX JOB...
//DETACH EXEC PGM=ADSIM012,
// PARM='...'
//STEPLIB DD DISP=SHR,DSN=SYS2.XPEDITER.V7R0M0.LOADLIB
```

PARM is where you specify the jobs to be cancelled and is entered as follows:

1. To detach a single job name TSOUSR1, enter:

   ```
   PARM='TSOUSR1'
   ```

2. To detach multiple job names TSOUSR1 and BATCHJB1, enter:

   ```
   PARM='TSOUSR1,BATCHJB1'
   ```

3. To detach all XPEDITER IMS dependent region tasks, enter:

   ```
   PARM='ALLXPED'
   ```

Note: If you do not know the job names, run the ADSIM012 program without a parameter string to obtain a list of each XPEDITER job or TSO session that is currently connected to the IMS control region. Alternatively, the operator can enter **/DIS A** on an IMS terminal to get the active job names. The job name for any XPEDITER test region running in TSO is the user's TSO ID.

The ADSIM012 program can also be run as a CLIST in the TSO foreground. To get status messages at the terminal, specify WTPMSG on the TSO PROFILE command as shown in the sample CLIST below:

```
PROFILE WTPMSG
CALL 'SYS2.XPEDITER.V7R0M0.LOADLIB(ADSIM012)' 'parm'
```

Where *parm* can be a single job name, multiple job names, or ALLXPED as described above.

Notes:

1. The ADSIM012 program will not detach XPEDITER regions running in XPBYPASS mode because of the risk of abending the control region with a U113. Regions running in XPBYPASS mode should be cancelled the same way that a non-XPEDITER region is cancelled.
2. To be effective, the ADSIM012 program must run in the same system where the IMS control region is running.
3. Because this method forces a termination of the XPEDITER test session, there may be an abend on the TSO session (in XPEDITER) while getting out of the test. This abend should not adversely affect any subsequent XPEDITER tests.
4. The ADSIM012 program will not detach XPEDITER regions in which the XPEDITER TCB handling the STIMER loop is rendered inactive because you pressed the **Attn**

key. However, normal IMS shutdown procedures (i.e., /STOP REG, /STOP REG ABDUMP, /STOP REG CANCEL) for these XPEDITER regions will work.

5. XPEDITER uses a STIMER loop within a separate monitor task. This disables the normal wait time timeout for the TSO region in which XPEDITER/IMS is active. The XPEDITER installer may want to enable the automatic XPEDITER/IMS region task detach facility. The member IMSDET in the SYS2.XPEDITER.V7R0M0.INSTALL library contains information about the installation of this facility.

Chapter 8. Debugging With DB2 Stored Procedures

XPEDITER/TSO, a testing and debugging product, supports the debugging of DB2 Stored Procedures using the DB2 Stored Procedure option. Operation of the product requires the Stored Procedure to reside on an OS/390 system. The client can be either local (OS/390) or remote.

DB2 Stored Procedure Requirements

XPEDITER/TSO DB2 Stored Procedure support requires:

- DB2 Version 5.1 or 6.1
- Workload Manager (WLM) managed Stored Procedures
- Installation of XPEDITER/TSO batch connect (refer to Chapter 2, “Step 12. Define VTAM APPLIDs for XPEDITER/TSO Batch Connect” on page 2-13 of the *XPEDITER/TSO and XPEDITER/IMS Installation Guide*).
- Installation of XPEDITER/TSO Stored Procedure support (refer to both Chapter 2 and Chapter 15 of the *XPEDITER/TSO and XPEDITER/IMS Installation Guide*)

Overview of DB2 Stored Procedure Support

To debug a Stored Procedure, you must first inform XPEDITER/TSO which Stored Procedure you want to debug. The Stored Procedure will be submitted with a predefined Application Environment for XPEDITER/TSO's use.

Next, the client application (the program that calls the Stored Procedure) must be started. The client can be local to OS/390 (such as batch, IMS, or CICS) or remote (such as Windows, Windows NT, OS/2, Unix, Powerbuilder, or Visual Basic).

After starting the client application, XPEDITER/TSO intercepts the request from DB2 to WLM to schedule a Stored Procedure to be executed. XPEDITER/TSO will then direct the Stored Procedure to use the previously submitted Stored Procedure with the XPEDITER/TSO application environment.

XPEDITER/TSO Stored Procedure support uses the Batch Connect feature to connect to the Stored Procedure. After the client application has made the call to the Stored Procedure, the Batch Connect session can be connected to. The source for the Stored Procedure will be displayed and execution will be paused at the entry to the Stored Procedure. From this point, all available XPEDITER/TSO debugging commands may be used, as with any debugging session.

Note: XPEDITER/Code Coverage also supports Stored Procedure programs.

After the debugging session is finished (due to the nature of the test session, you will be positioned in the XPEDITER/TSO LOG with ***TEST COMPLETED*** displayed), there are two ways to end the session.

1. The normal method is to issue the GO command. This will instruct XPEDITER/TSO to execute the epilog portion of the application. The Stored Procedure will return requested values and return codes to the invoking (CLIENT) program.

2. You may also issue the EXIT command, which instructs XPEDITER/TSO to end the test immediately. Be aware that, because this action prematurely ends the test (the application epilog code is not executed), this is treated as an **ABEND** (**AB**normal **END**). If the EXIT command is issued at any point other than after the XPEDITER **TEST COMPLETED** message has been received, any DB2 table updates may be rolled back during this period and the Stored Procedure will **NOT** return requested values and return codes to the invoking (CLIENT) program.

When you set up the debugging session, you will be required to enter how many times the Stored Procedure will be intercepted by XPEDITER/TSO (this is specified in the ***Maximum Number of Tests*** field). If you entered **1**, the debugging session will now end. If you entered a number greater than **1**, XPEDITER/TSO will wait for the next call to the Stored Procedure to restart the debugging session.

Starting an XPEDITER/TSO DB2 Stored Procedure Debugging Session

XPEDITER/TSO consists of specific menus and screens accessed through ISPF. During your debugging session, XPEDITER/TSO screens are viewed on a TSO terminal.

Before starting the session, prepare your programs (precompile, compile and link edit) with the Compuware Shared Services (CSS) language processor. You can use the XPEDITER/TSO online facilities to prepare your programs. Refer to Chapter 4, "Preparing Your Programs" on page 4-3 of the *XPEDITER/TSO and XPEDITER/IMS COBOL User's Guide* for additional information. For more information about Compuware's Shared Services, refer to the Compuware *Enterprise Common Components Installation and Customization Guide*.

Once you have a valid load module and DDIO dataset for the Stored Procedure, log on to TSO and invoke XPEDITER/TSO. Use the procedures that are in effect at your site. These procedures will include the following steps:

1. From the XPEDITER/TSO Primary Menu (Figure 8-1) select option **4** (Stored Procedures) and press Enter.

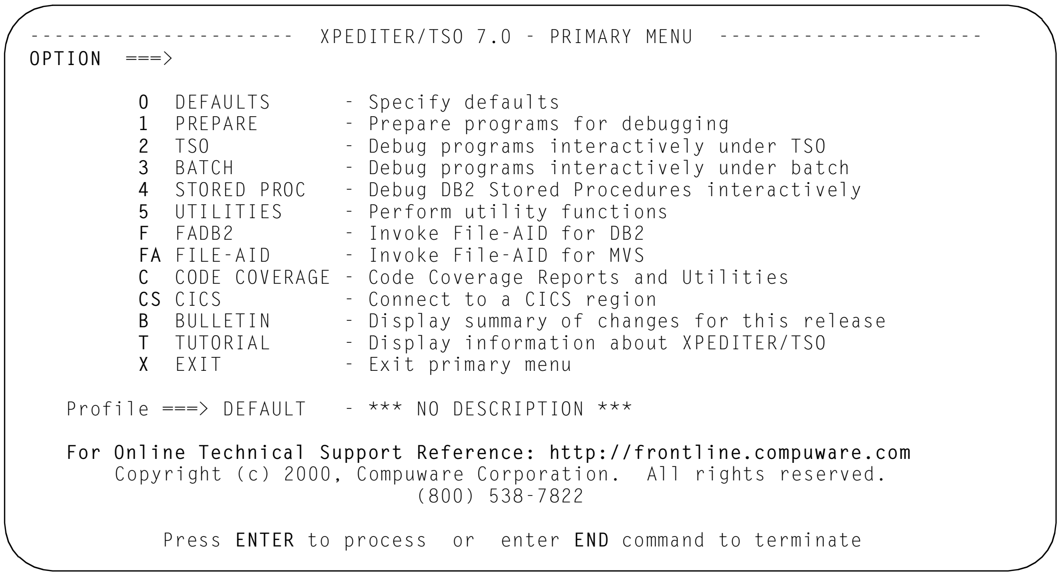

```
---------------------  XPEDITER/TSO 7.0 - PRIMARY MENU  ---------------------
OPTION  ===>

        0  DEFAULTS      - Specify defaults
        1  PREPARE       - Prepare programs for debugging
        2  TSO           - Debug programs interactively under TSO
        3  BATCH         - Debug programs interactively under batch
        4  STORED PROC   - Debug DB2 Stored Procedures interactively
        5  UTILITIES     - Perform utility functions
        F  FADB2         - Invoke File-AID for DB2
        FA FILE-AID      - Invoke File-AID for MVS
        C  CODE COVERAGE - Code Coverage Reports and Utilities
        CS CICS          - Connect to a CICS region
        B  BULLETIN      - Display summary of changes for this release
        T  TUTORIAL      - Display information about XPEDITER/TSO
        X  EXIT          - Exit primary menu

   Profile ===> DEFAULT   - *** NO DESCRIPTION ***

   For Online Technical Support Reference: http://frontline.compuware.com
       Copyright (c) 2000, Compuware Corporation.  All rights reserved.
                               (800) 538-7822

           Press ENTER to process  or  enter END command to terminate
```

Figure 8-1. XPEDITER/TSO Primary Menu

Note: The XPEDITER/TSO Process DB2 Stored Procedures Test screen (Figure 8-2 on page 8-4) is displayed.

2. On the Process DB2 Stored Procedure Test screen (Figure 8-2 on page 8-4), type **SETUP** on the command line and press Enter to display the XPEDITER/TSO Setup Menu screen (Figure 8-3 on page 8-5).
3. Specify an **A** (All) in the "***OPTION***" field of the Setup Menu screen and press **Enter** to display the Base Product Datasets screen. Continue to press **Enter** to access additional screens and review all of your setup selections. Note the following:
   - Load libraries—user program libraries allocated as STEPLIB. Verify that the order of concatenation is correct. Help libraries are also listed on this screen.
   - DDIO Dataset Files—the library name should be the dataset name of the DD CWPDDIO in your XPEDITER compile step. The field designations are User Libraries and Installation Libraries.
   - Test script libraries.
   - Test session LOG dataset.
   - Test session script dataset.
   - DB2 system names and DSNLOAD datasets.
   - Code Coverage Repository dataset name, System name, and Test IDs (if installed at your site.

   Note: As you view the individual screens, you may encounter the message:

   ```
   Changes made to this list override installed defaults
   ```
4. If everything is correct, press PF3 from the Setup Menu to return to the test screen (refer to Figure 8-2 on page 8-4).
5. Fill in the Stored Procedure name, Client USER ID, and DB2 Subsystem name that you wish to debug.
6. Set the "*Specify Execute JCL*" field to **N** (No) to make XPEDITER retrieve the JCL defined for the Stored Procedure. Set "*Specify Execute JCL*" to **Y** (Yes) if you wish to specify your own JCL to execute the Stored Procedure. When specifying **Y**, an additional screen will be displayed where you must enter the dataset that contains the JCL (this JCL must execute the WLM server).
7. Convert and Submit the JCL: The Select Job Step screen (Figure 8-5 on page 8-7) is used to specify how you want each job step to execute (in interactive or unattended mode), and if you want Code Coverage to be active or not. When a job step is selected for interactive debugging, the source for that step is displayed at your terminal.

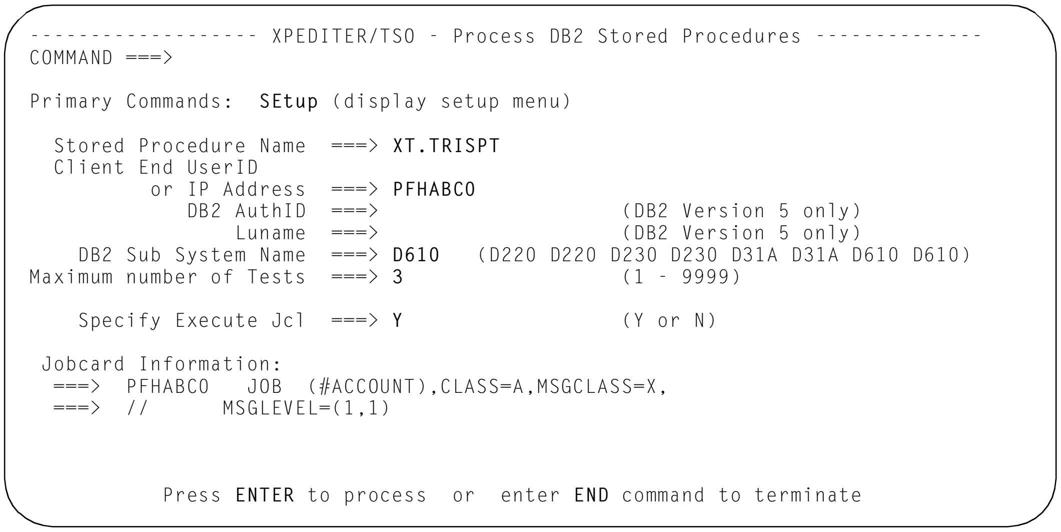
   When a step is selected to run in unattended mode, you cannot interact with the step from your terminal. If you want to debug the step, XPEDITER debugging commands must be read from a test script and the output results from the test session are written to the LOG.

   You can submit the job from this screen or you can edit the JCL. When you submit the job, the batch connect intelligent scanner automatically converts each step to execute in the specified mode (interactive or unattended).
8. Connect to the Job: If you submit the job with the RUN command, you are automatically connected to the job steps that were specified to execute in interactive mode. If you submit the job with the SUBMIT command, you must use either the CONNECT or STATUS command to connect your terminal to the job steps selected for interactive debugging.
9. Edit the JCL: Optionally, you can access the Edit Facility from the Process Execute JCL or Select Job Step screen to view and confirm the JCL conversion, or to make changes to the JCL.
10. Once the debugging session is submitted, XPEDITER/TSO waits for the Stored Procedure to be called by the client application. Once the call has been made, the XPEDITER/TSO session can be connected to.

11. View the XPEDITER/TSO source display on the TSO terminal.
12. Debug the Stored Procedure using any of the XPEDITER/TSO debugging commands.
13. At the end of the test type **GO** or **EXIT**. EXIT is a premature end to the test and will cause an abend. DB2 updates may be rolled back during this period.
14. If you enter a number greater than **1** in the "*Maximum number of tests*" field, XPEDITER/TSO will wait until the next time the Stored Procedure is called, and then return you again to the source.

Debugging a DB2 Stored Procedure

From the XPEDITER/TSO primary menu, select OPTION **4** (STORED PROC) to debug a DB2 Stored Procedure. Press Enter and the XPEDITER/TSO Process DB2 Stored Procedures Test screen shown in Figure 8-2 is displayed.



```
------------------ XPEDITER/TSO - Process DB2 Stored Procedures -------------
COMMAND ===>

Primary Commands:  SEtup (display setup menu)

  Stored Procedure Name  ===> XT.TRISPT
  Client End UserID
          or IP Address  ===> PFHABCO
             DB2 AuthID  ===>                  (DB2 Version 5 only)
                 Luname  ===>                  (DB2 Version 5 only)
    DB2 Sub System Name  ===> D610   (D220 D220 D230 D230 D31A D31A D610 D610)
Maximum number of Tests  ===> 3                (1 - 9999)

    Specify Execute Jcl  ===> Y                (Y or N)

 Jobcard Information:
  ===>  PFHABCO   JOB  (#ACCOUNT),CLASS=A,MSGCLASS=X,
  ===>  //       MSGLEVEL=(1,1)



         Press ENTER to process  or  enter END command to terminate
```

Figure 8-2. XPEDITER/TSO Process DB2 Stored Procedures Test Screen

After the required information is entered, the JCL needed to execute the Stored Procedure must be submitted.

The SEtup command displays the Setup Menu screen (Figure 8-3 on page 8-5) from which you can select screens for specifying load libraries, DDIO datasets, and so on.

The fields on the Stored Procedures Test screen are:

Stored Procedure Name	Used to identify the OS/390 DB2 Stored Procedure to be debugged using XPEDITER/TSO. If you are using DB2 Version 6.1 or higher, it also contains the ***Schema*** name (prefixed to the Stored Procedure name with a period).
Client End UserID or IP Address	The ID of the user that is executing the client program calling this Stored Procedure. Users (other than the one specified) calling this Stored Procedure will not be trapped by XPEDITER/TSO. Field is required.
DB2 AuthID	The authorization ID for the client application. This field is used to determine the correct Stored Procedure to load. Required for DB2 Version 5.1 only.
Luname	The name of the system where the client application is running. This field is used to determine the cor-

	rect Stored Procedure to load. Required for DB2 Version 5.1 only.
DB2 Sub System Name	The name of the DB2 Subsystem which this Stored Procedure is running under.
Maximum number of Tests	This field indicates how many times the Stored Procedure will be intercepted by XPEDITER/TSO. If a Stored Procedure is called multiple times and this field is set to **1**, only the first iteration will be trapped by XPEDITER/TSO. Subsequent calls will run outside of XPEDITER/TSO.
Specify Execute JCL	Enter either **Y** (Yes) or **N** (No). N indicates that XPEDITER/TSO will retrieve the JCL from DB2 to run the Stored Procedure. Y indicates that you will provide the JCL to run the Stored Procedure. A subsequent panel will be displayed to enter the dataset name of the JCL.
Jobcard Information	Enter the jobcard information to be used for the Stored Procedure test that is executed. If **N** was entered in the "***Specify Execute Jcl***" field, the JOBCARD will be used for the retrieved Stored Procedure JCL when the test is submitted.

Accessing the XPEDITER/TSO Setup Menu Screen

If you type **SEtup** on the command line of the Process DB2 Stored Procedures screen and press Enter, the XPEDITER/TSO Setup Menu Screen shown in Figure 8-3 is displayed.

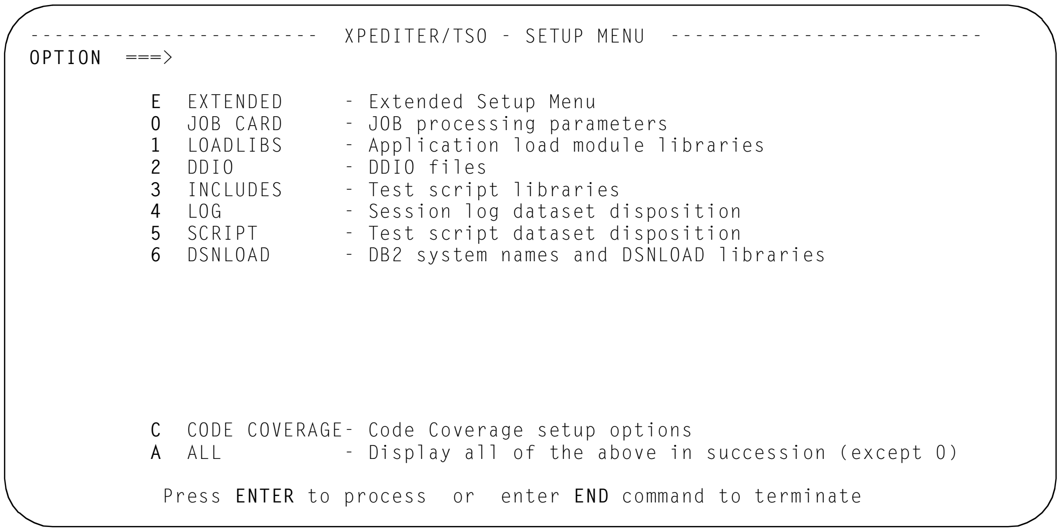

```
------------------------  XPEDITER/TSO - SETUP MENU  -------------------------
OPTION  ===>

          E  EXTENDED      - Extended Setup Menu
          0  JOB CARD      - JOB processing parameters
          1  LOADLIBS      - Application load module libraries
          2  DDIO          - DDIO files
          3  INCLUDES      - Test script libraries
          4  LOG           - Session log dataset disposition
          5  SCRIPT        - Test script dataset disposition
          6  DSNLOAD       - DB2 system names and DSNLOAD libraries






          C  CODE COVERAGE- Code Coverage setup options
          A  ALL           - Display all of the above in succession (except 0)

           Press ENTER to process  or  enter END command to terminate
```

Figure 8-3. XPEDITER/TSO Setup Menu Screen

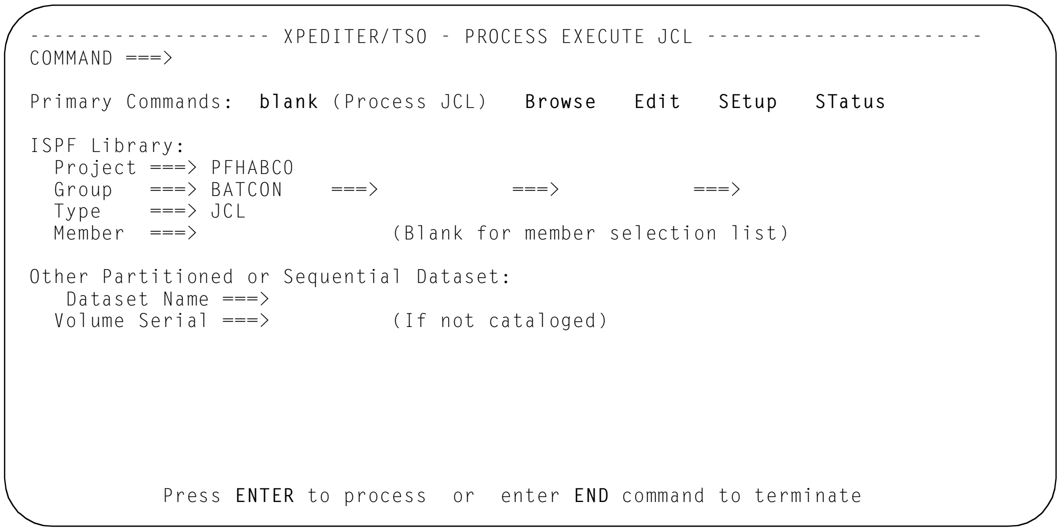
To access the screens containing Load Libraries, DDIO dataset, etc., type an **A** (All) on the command line of the XPEDITER/TSO Setup Menu Screen and press Enter.

Each of the successive screens (Test script libraries, Test Session LOG dataset, DSNLOAD datasets, etc.) are accessible (in consecutive order) by continuing to press Enter.

Specifying the JCL

If you enter a **Y** in the "*Specify Execute JCL*" field of the XPEDITER/TSO Process DB2 Stored Procedure screen, Figure 8-4 on page 8-6 will be displayed. From the Process

Execute JCL screen, select the name of your dataset that contains the JCL to execute the Stored Procedure. Type this name in the "*Dataset Name*" field and press Enter. This JCL must execute the WLM server.



```
-------------------- XPEDITER/TSO - PROCESS EXECUTE JCL -----------------------
COMMAND ===>

Primary Commands:  blank (Process JCL)   Browse   Edit   SEtup   STatus

ISPF Library:
  Project ===> PFHABCO
  Group   ===> BATCON    ===>          ===>          ===>
  Type    ===> JCL
  Member  ===>               (Blank for member selection list)

Other Partitioned or Sequential Dataset:
   Dataset Name ===>
  Volume Serial ===>         (If not cataloged)




           Press ENTER to process  or  enter END command to terminate
```

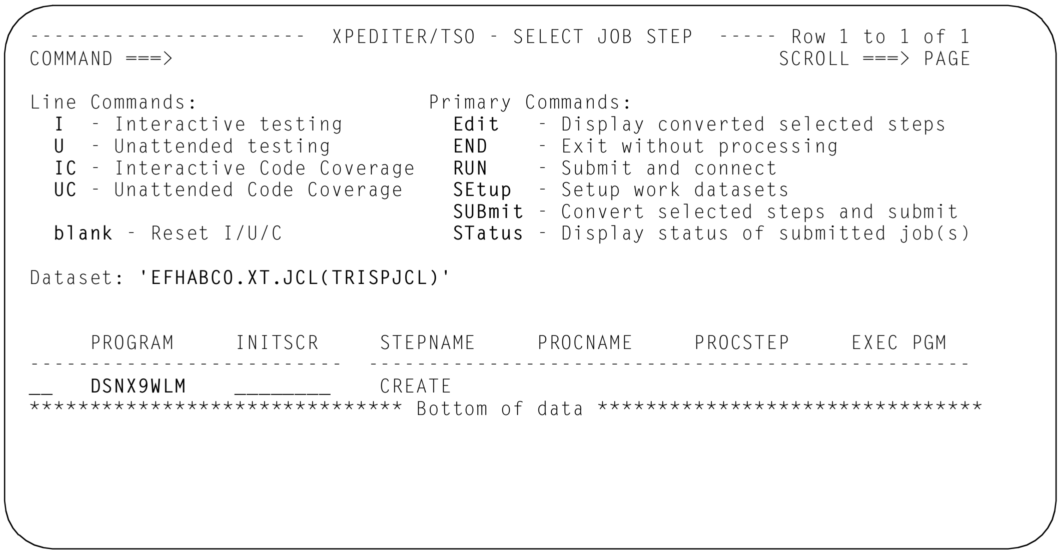
Figure 8-4. Process Execute JCL Screen

There are two steps you must perform on the Process Execute JCL screen:

1. Enter the name of the dataset containing your execution JCL, either in the "***ISPF Library***" field or in the "***Other Partitioned or Sequential Dataset***" field.
2. Use one of the following primary commands on the command line:

blank	Leave the command line blank to scan the specified JCL and extract the job step information. If the JCL is a procedure, it is expanded prior to scanning. A progress message is displayed during processing. When processing is complete, the Select Job Step screen shown in Figure 8-5 on page 8-7 is displayed.
Browse	Invokes the ISPF browse facility and displays the specified JCL.
Edit	Invokes the ISPF edit facility and displays the specified JCL. You can edit the JCL. Refer to Chapter 4, "Editing the JCL" on page 4-22 for more information.
SEtup	Accesses the Test Setup Menu that will let you access the job card information needed to process JCL PROCs. Refer to Appendix C, "Specifying Setup Options" on page C-1 for more information.
STatus	Displays the status of any submitted job(s). You can connect to a job from the Status screen.

When JCL processing is completed, the job steps in the specified JCL are displayed on the Select Job Step screen shown in Figure 8-5 on page 8-7.



```
----------------------  XPEDITER/TSO - SELECT JOB STEP  ----- Row 1 to 1 of 1
COMMAND ===>                                                  SCROLL ===> PAGE

Line Commands:                   Primary Commands:
  I  - Interactive testing         Edit   - Display converted selected steps
  U  - Unattended testing          END    - Exit without processing
  IC - Interactive Code Coverage   RUN    - Submit and connect
  UC - Unattended Code Coverage    SEtup  - Setup work datasets
                                   SUBmit - Convert selected steps and submit
  blank - Reset I/U/C              STatus - Display status of submitted job(s)

Dataset: 'EFHABCO.XT.JCL(TRISPJCL)'


     PROGRAM     INITSCR     STEPNAME     PROCNAME     PROCSTEP     EXEC PGM
--------------------------  ------------------------------------------------
__   DSNX9WLM    ________    CREATE
****************************** Bottom of data *******************************
```

Figure 8-5. XPEDITER/TSO Select Job Step Screen

The Select Job Step screen is used in two ways:

1. Use the I, IC, U, or UC line command to specify how you want each job step to execute: in interactive or unattended mode. The JCL will be converted to run with XPEDITER/TSO in the specified execution mode. Refer to Chapter 4, "JCL Conversion" on page 4-24 for information about JCL conversion.
2. Use the RUN or SUBMIT primary command to convert and submit the JCL, the EDIT command to view or edit the JCL, the SETUP command to access and change your job card information, the STATUS command to display the status of submitted jobs, or the END command to exit.

The Line commands are defined as follows:

I (Interactive) testing	Selects the step to run in interactive mode.
U (Unattended) testing	Selects the step to run in unattended mode.
IC (Interactive Code Coverage)	Selects the step to run in interactive mode (with Code Coverage active).
UC (Unattended Code Coverage)	Selects the step to run in unattended mode (with Code Coverage active).
blank (Reset I/U/C)	Blank the line command area and press **Enter** to remove an I, IC, U, or UC line command set on a job step.

The Primary commands are defined as follows:

Edit Accesses the ISPF edit facility. The JCL is automatically converted and displayed on the ISPF edit screen. You can view and confirm the JCL conversion and make additional modifications to the converted JCL. However, the changes made to the converted JCL will not be saved when you select CANCEL or PF3 (END) from the edit screen. Refer to Chapter 4, "Editing the JCL" on page 4-22 for more information and for an example of the ISPF edit screen on page 4-23.

Note: You can submit the JCL from the edit screen using the RUN or SUBMIT command.

END Exits without converting the JCL, saving any modifications, or submitting the job, and returns you to the Process Execute JCL screen.

RUN Converts the steps selected by the I and U line commands to predefined XPEDITER/TSO steps and submits the JCL. When the job processes successfully, you are automatically connected to the job steps selected to run in interactive mode and the Source screen is displayed.

If the job is a long-running job, the Connect Status screen is displayed showing the job status.

Note: The terminal cannot be used while the job is running.

If the JCL is already converted, the RUN command (with a DSNAME) can be entered on any screen except an XPEDITER test session screen. Refer to the *XPEDITER/TSO and XPEDITER/IMS Reference Manual* for more information about the RUN command.

SEtup Displays the Setup Menu from which you can select to view and change the job card and library information, such as the DDIO file (XPSL000n) and the SCRIPT file (XINCLUDE). Refer to Appendix C, "Specifying Setup Options" on page C-1 for more information.

SUBmit Converts the steps selected by the I, IC, U, and UC commands to XPEDITER/TSO steps and submits the job. A job submitted with the SUBMIT command is not automatically connected. To connect to a job submitted with the SUBMIT command, you must use the CONNECT or STATUS command. Refer to Chapter 4, "Connecting to a Job" on page 4-21 for more information regarding the CONNECT or STATUS command.

Note: While the job is running, you can continue using your terminal.

STatus Displays the status of a job. You can connect to the job directly from the Status screen.

The fields displayed on the Select Job Step screen are defined as follows:

Dataset This field is pre-filled with the dataset name of the JCL being processed.

PROGRAM The name of the program to be tested. The PROGRAM name does not necessarily match the EXEC PGM name.

INITSCR The member name of the script in the INCLUDE library specified on the Setup panel. The INITSCR field can be typed over to specify a test script member, which can then be processed at the beginning of a debugging session.

STEPNAME The job step name.

PROCNAME The in-stream or cataloged procedure name.

PROCSTEP The step name within the called procedure.

EXEC PGM The name of the EXEC program that is executed for the step. XPTSO if I (Interactive) or IC (Interactive Code Coverage) was specified for the step. XPBATCH if U (Unattended) or UC (Unattended Code Coverage) was selected for the step. The field is left blank if the name is the same as the one entered in the PROGRAM field.

Connecting to a Job

When you use the RUN command to submit the job, the steps that are selected for interactive debugging are automatically displayed at your terminal.

When you use the SUBMIT command to submit the job, there are two ways to connect to a job—the CONNECT command and the STATUS command:

1. CONNECT Command: You can use the CONNECT command on any screen (except the source display) to connect a VTAM terminal to a job submitted through XPEDITER/TSO's Batch Connect facility. You can connect to a job with multiple steps or to a single step job. For information on the CONNECT command syntax, refer to the *XPEDITER/TSO and XPEDITER/IMS Reference Manual*.

2. STATUS command: The STATUS command is used to display the Status screen containing a list of the active/inactive jobs in the system. The STATUS command can be entered from any screen.

The ATTACH line command on the Status screen is used to connect to a job and display the source of each job step for which the I (Interactive) command was specified. A message is displayed notifying you that the job step selected for testing is executing.

Editing the JCL

There are two points at which the Batch Connect facility lets you edit your JCL.

1. Primary editing is available by entering the **Edit** primary command on the Process Execute JCL screen (Figure 8-4 on page 8-6). An ISPF edit session is invoked and the specified JCL is displayed. If your site security permits, changes will be saved to the original JCL when the edit session ends.
2. Secondary editing is available by entering the Edit primary command on the Select Job Step screen # 2 (refer to Figure 8-6). An ISPF edit session is invoked and a temporary copy (45 lines) of JCL as shown in Figure 8-7 on page 8-10 is displayed. JCL statements for those steps selected for testing, using the I (Interactive) or U (Unattended) line commands, are already converted when the edit screen is displayed.

   After any editing is completed, you can submit the job from the XPEDITER/TSO Select Job Step screen # 2 (Figure 8-6) with the **RUN** or **SUB**mit command.

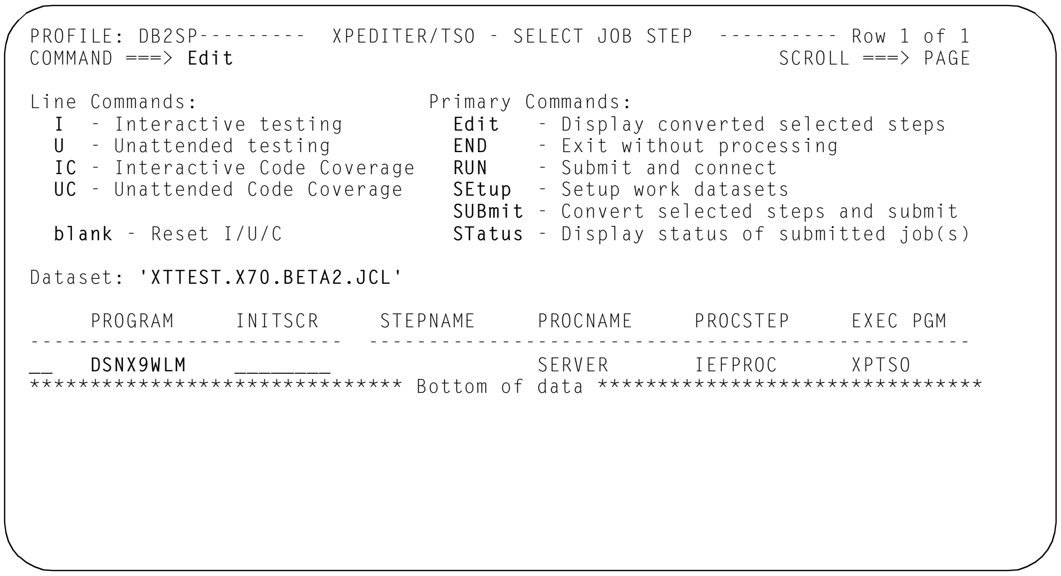
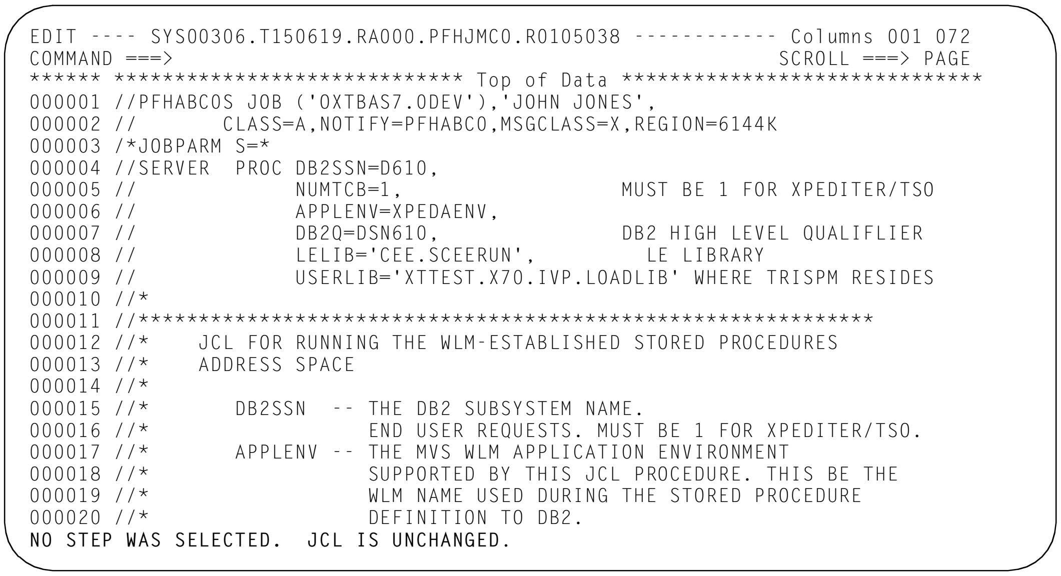

```
PROFILE: DB2SP---------  XPEDITER/TSO - SELECT JOB STEP  ---------- Row 1 of 1
COMMAND ===> Edit                                              SCROLL ===> PAGE

Line Commands:                   Primary Commands:
  I  - Interactive testing         Edit   - Display converted selected steps
  U  - Unattended testing          END    - Exit without processing
  IC - Interactive Code Coverage   RUN    - Submit and connect
  UC - Unattended Code Coverage    SEtup  - Setup work datasets
                                   SUBmit - Convert selected steps and submit
  blank - Reset I/U/C              STatus - Display status of submitted job(s)

Dataset: 'XTTEST.X70.BETA2.JCL'

     PROGRAM     INITSCR     STEPNAME     PROCNAME     PROCSTEP     EXEC PGM
-------------------------  ----------------------------------------------------
__   DSNX9WLM    ________                 SERVER       IEFPROC      XPTSO
****************************** Bottom of data *******************************
```

Figure 8-6. XPEDITER/TSO Select Job Step Screen # 2



```
EDIT ---- SYS00306.T150619.RA000.PFHJMC0.R0105038 ----------- Columns 001 072
COMMAND ===>                                                  SCROLL ===> PAGE
****** ***************************** Top of Data ******************************
000001 //PFHABC0S JOB ('OXTBAS7.ODEV'),'JOHN JONES',
000002 //        CLASS=A,NOTIFY=PFHABC0,MSGCLASS=X,REGION=6144K
000003 /*JOBPARM S=*
000004 //SERVER  PROC DB2SSN=D610,
000005 //              NUMTCB=1,                  MUST BE 1 FOR XPEDITER/TSO
000006 //              APPLENV=XPEDAENV,
000007 //              DB2Q=DSN610,               DB2 HIGH LEVEL QUALIFLIER
000008 //              LELIB='CEE.SCEERUN',         LE LIBRARY
000009 //              USERLIB='XTTEST.X70.IVP.LOADLIB' WHERE TRISPM RESIDES
000010 //*
000011 //*************************************************************
000012 //*    JCL FOR RUNNING THE WLM-ESTABLISHED STORED PROCEDURES
000013 //*    ADDRESS SPACE
000014 //*
000015 //*       DB2SSN  -- THE DB2 SUBSYSTEM NAME.
000016 //*                  END USER REQUESTS. MUST BE 1 FOR XPEDITER/TSO.
000017 //*       APPLENV -- THE MVS WLM APPLICATION ENVIRONMENT
000018 //*                  SUPPORTED BY THIS JCL PROCEDURE. THIS BE THE
000019 //*                  WLM NAME USED DURING THE STORED PROCEDURE
000020 //*                  DEFINITION TO DB2.
NO STEP WAS SELECTED.  JCL IS UNCHANGED.
```

Figure 8-7. Sample Edit JCL Screen

Debugging the Client Application <u>*AND*</u> the DB2 Stored Procedure

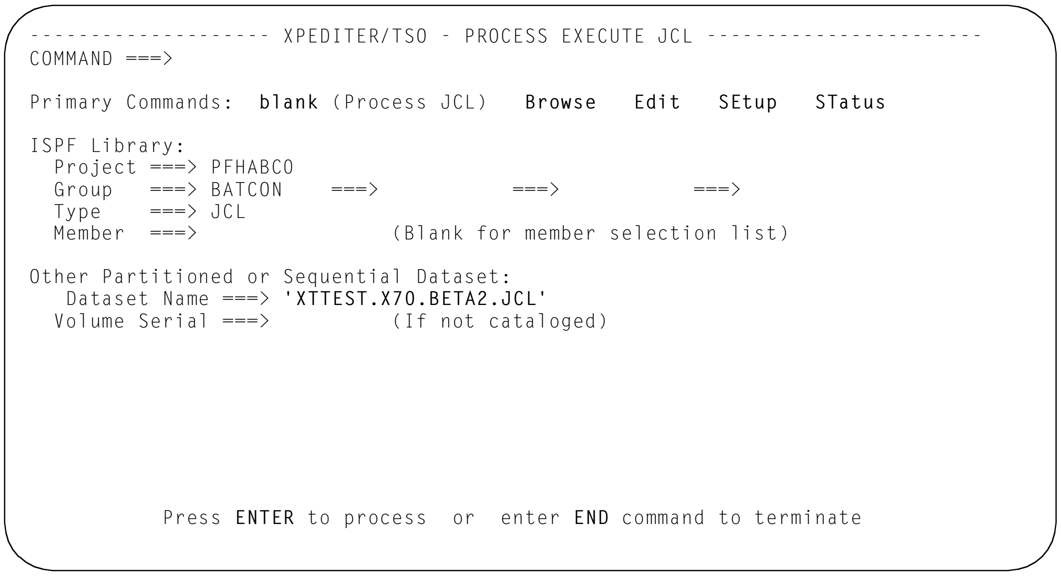
The following instructions pertain to a specialized technique that allows for the debugging of a *Client application* (i.e., a DB2 batch program) which calls a DB2 Stored Procedure, while using a single TSO userid.

Note: This technique will only work if the *Client application* is submitted as a batch program set up to run with XPEDITER/TSO Batch Connect.

CAUTION:
To become familiar with the process outlined below, it may be beneficial to step through the process a few times using the Stored Procedure Initial Verification Procedure (IVP) found in the *XPEDITER/TSO and XPEDITER/IMS Installation Guide***.**

Note the following instructions:

1. Access the XPEDITER/TSO Process Execute JCL screen (Figure 8-8) by entering a **Y** in the "***Specify Execute JCL***" field of the XPEDITER/TSO Process DB2 Stored Procedure screen (Figure 8-2 on page 8-4).

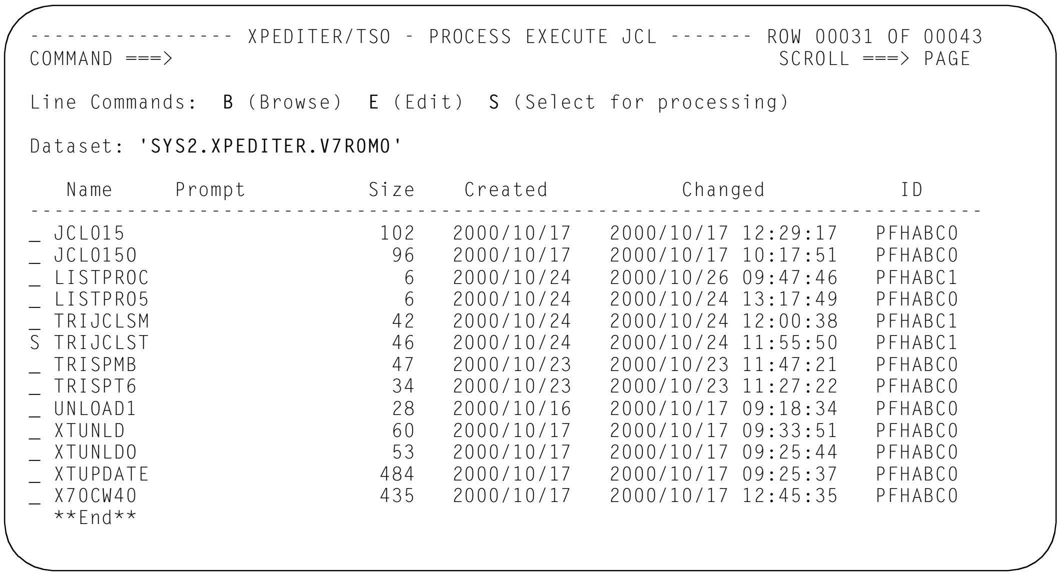


```
-------------------- XPEDITER/TSO - PROCESS EXECUTE JCL ----------------------
COMMAND ===>

Primary Commands:  blank (Process JCL)   Browse   Edit   SEtup   STatus

ISPF Library:
  Project ===> PFHABCO
  Group   ===> BATCON    ===>          ===>          ===>
  Type    ===> JCL
  Member  ===>               (Blank for member selection list)

Other Partitioned or Sequential Dataset:
   Dataset Name ===> 'XTTEST.X70.BETA2.JCL'
  Volume Serial ===>         (If not cataloged)






           Press ENTER to process  or  enter END command to terminate
```

Figure 8-8. XPEDITER/TSO Process Execute JCL Screen

2. Type the name of your pre-defined dataset in the "***Dataset Name***" field of the Process Execute JCL screen and press Enter to access the **second** Process Execute JCL screen (Figure 8-9 on page 8-12). If you wish to have a dataset name provided for this example, use the dataset name of the XPEDITER/TSO SAMPLIB. The default name is:

   `'SYS2.XPEDITER.V7R0M0.SAMPLIB'`



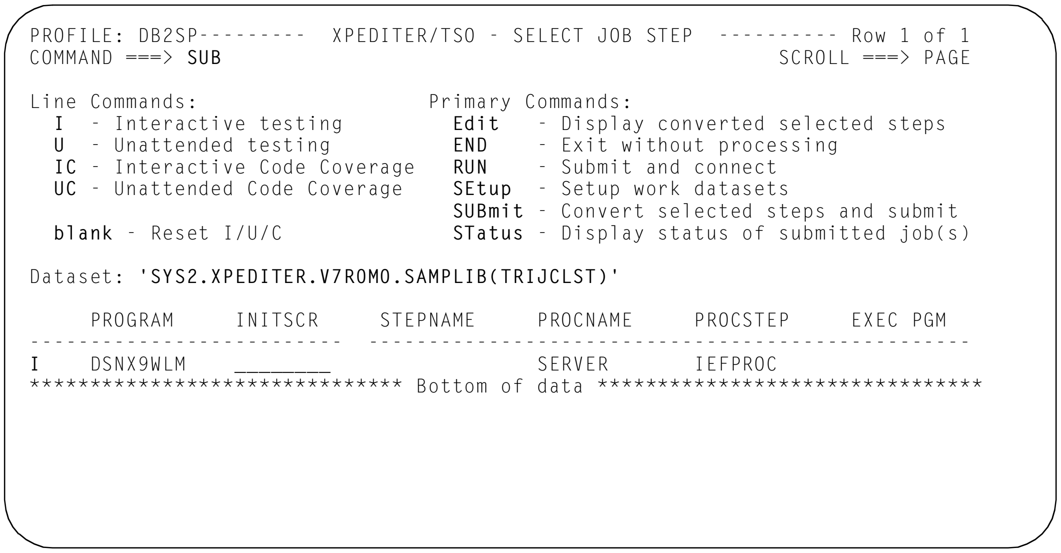
```
---------------- XPEDITER/TSO - PROCESS EXECUTE JCL ------- ROW 00031 OF 00043
COMMAND ===>                                                 SCROLL ===> PAGE

Line Commands:  B (Browse)  E (Edit)  S (Select for processing)

Dataset: 'SYS2.XPEDITER.V7R0M0'

   Name     Prompt          Size    Created          Changed          ID
------------------------------------------------------------------------------
_ JCL015                     102   2000/10/17   2000/10/17 12:29:17   PFHABC0
_ JCL0150                     96   2000/10/17   2000/10/17 10:17:51   PFHABC0
_ LISTPROC                     6   2000/10/24   2000/10/26 09:47:46   PFHABC1
_ LISTPRO5                     6   2000/10/24   2000/10/24 13:17:49   PFHABC0
_ TRIJCLSM                    42   2000/10/24   2000/10/24 12:00:38   PFHABC1
S TRIJCLST                    46   2000/10/24   2000/10/24 11:55:50   PFHABC1
_ TRISPMB                     47   2000/10/23   2000/10/23 11:47:21   PFHABC0
_ TRISPT6                     34   2000/10/23   2000/10/23 11:27:22   PFHABC0
_ UNLOAD1                     28   2000/10/16   2000/10/17 09:18:34   PFHABC0
_ XTUNLD                      60   2000/10/17   2000/10/17 09:33:51   PFHABC0
_ XTUNLD0                     53   2000/10/17   2000/10/17 09:25:44   PFHABC0
_ XTUPDATE                   484   2000/10/17   2000/10/17 09:25:37   PFHABC0
_ X70CW40                    435   2000/10/17   2000/10/17 12:45:35   PFHABC0
  **End**
```

Figure 8-9. Second Process Execute JCL Screen

3. Scroll down (PF8) to the member name which has been defined for the Stored Procedure. For the continuity of this example, the member name is *TRIJCLST*. Type an **S** in the field to the left of TRIJCLST and press Enter. If you do not receive any error messages, you should have returned to the Select Job Step Screen (similar to Figure 8-10, Select Job Step Screen # 3).

   The "***Dataset***" field should now include the member name (TRIJCLST), enclosed in parentheses.

4. Type **SUBmit** on the Command line of the XPEDITER/TSO Select Job Step Screen # 3 (as shown in Figure 8-10). Type an **I** in the field immediately to the left of the "***PROGRAM***" field of the Select Job Step Screen # 3 (Figure 8-10). Press Enter to submit the DB2 Stored Procedure. You should receive an online message at the bottom of your screen stating that your job was submitted.

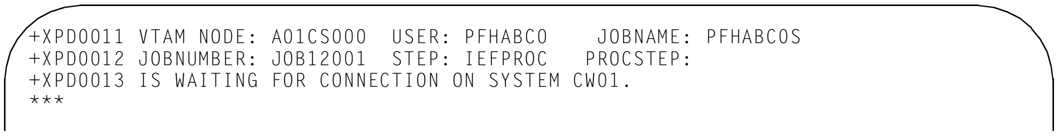


```
PROFILE: DB2SP---------  XPEDITER/TSO - SELECT JOB STEP  ---------- Row 1 of 1
COMMAND ===> SUB                                             SCROLL ===> PAGE

Line Commands:                  Primary Commands:
  I  - Interactive testing        Edit   - Display converted selected steps
  U  - Unattended testing         END    - Exit without processing
  IC - Interactive Code Coverage  RUN    - Submit and connect
  UC - Unattended Code Coverage   SEtup  - Setup work datasets
                                  SUBmit - Convert selected steps and submit
  blank - Reset I/U/C             STatus - Display status of submitted job(s)

Dataset: 'SYS2.XPEDITER.V7R0M0.SAMPLIB(TRIJCLST)'

     PROGRAM     INITSCR     STEPNAME     PROCNAME     PROCSTEP     EXEC PGM
--------------------------  -------------------------------------------------
I    DSNX9WLM    ________                 SERVER       IEFPROC
****************************** Bottom of data *******************************
```

Figure 8-10. XPEDITER/TSO Select Job Step # 3

5. Press Enter and you will view a four line message similar to the message displayed in Figure 8-11 on page 8-13. If you do not receive the "Waiting for Connection" message, you will view a message at the bottom of your screen stating:

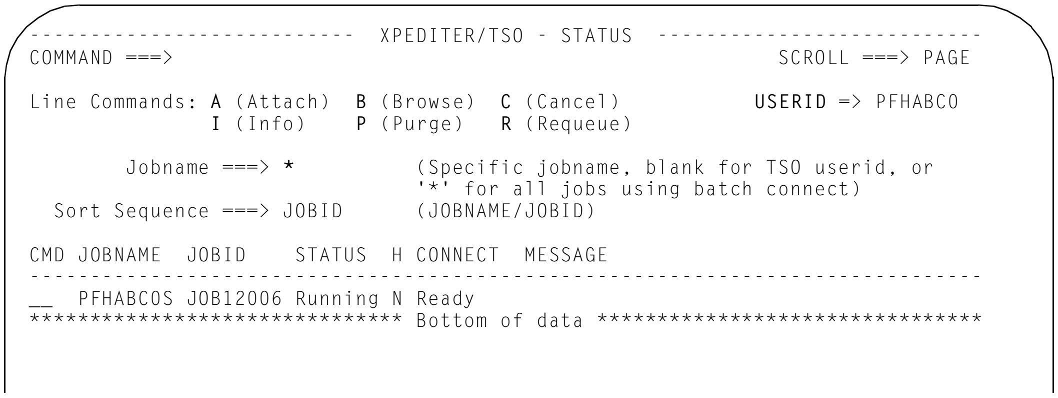
```
JCL HAS BEEN MODIFIED TO DEBUG WITH XPEDITER
```



```
+XPD0011 VTAM NODE: A01CS000  USER: PFHABCO    JOBNAME: PFHABCOS
+XPD0012 JOBNUMBER: JOB12001  STEP: IEFPROC   PROCSTEP:
+XPD0013 IS WAITING FOR CONNECTION ON SYSTEM CW01.
***
```

Figure 8-11. Stored Procedure 'Waiting for Connection' Message

6. Press Enter to remove the messages from your screen. You should remain in the Select Job Step screen # 3 (Figure 8-10 on page 8-12).
7. Type **STatus** on the Command line of the Select Job Step screen and press Enter to access the XPEDITER/TSO Status screen # 1 shown in Figure 8-12.



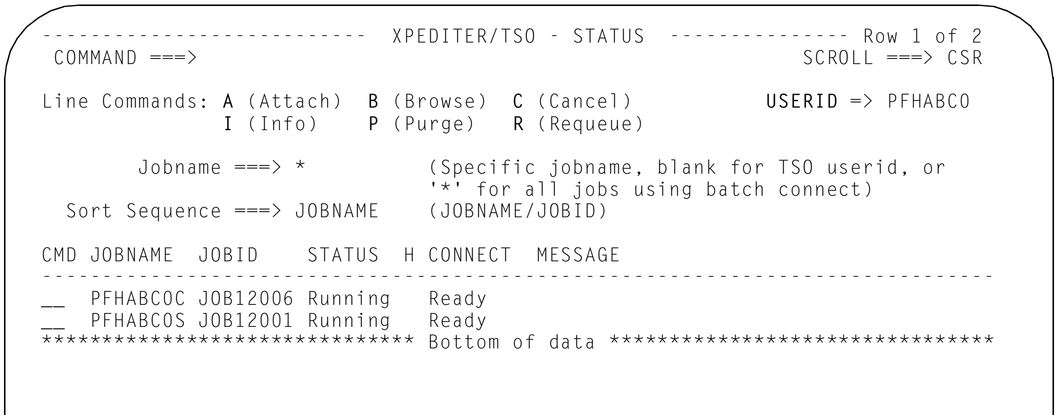
```
---------------------------  XPEDITER/TSO - STATUS  ---------------------------
COMMAND ===>                                                  SCROLL ===> PAGE

Line Commands: A (Attach)  B (Browse)  C (Cancel)           USERID => PFHABCO
               I (Info)    P (Purge)   R (Requeue)

       Jobname ===> *          (Specific jobname, blank for TSO userid, or
                               '*' for all jobs using batch connect)
  Sort Sequence ===> JOBID     (JOBNAME/JOBID)

CMD JOBNAME  JOBID    STATUS  H CONNECT  MESSAGE
-------------------------------------------------------------------------------
__  PFHABCOS JOB12006 Running N Ready
****************************** Bottom of data *******************************
```

Figure 8-12. XPEDITER/TSO Status Screen # 1 (Stored Procedure)

8. Type an asterisk **(*)** in the "***Jobname***" field of the XPEDITER/TSO Status Screen # 1 (Figure 8-12) and press Enter to refresh the Status screen and view the status of the *Stored Procedure* JOB. A single line of data is displayed on Status Screen # 1 (Figure 8-12). This line displays the Stored Procedure Jobname in the "***JOBNAME***" field and a valid JOBID number. The "***CONNECT***" field should indicate that the Stored Procedure is *Ready* to connect.

   Note: The default security for Batch Connect requires that the jobname must begin with your USERID.

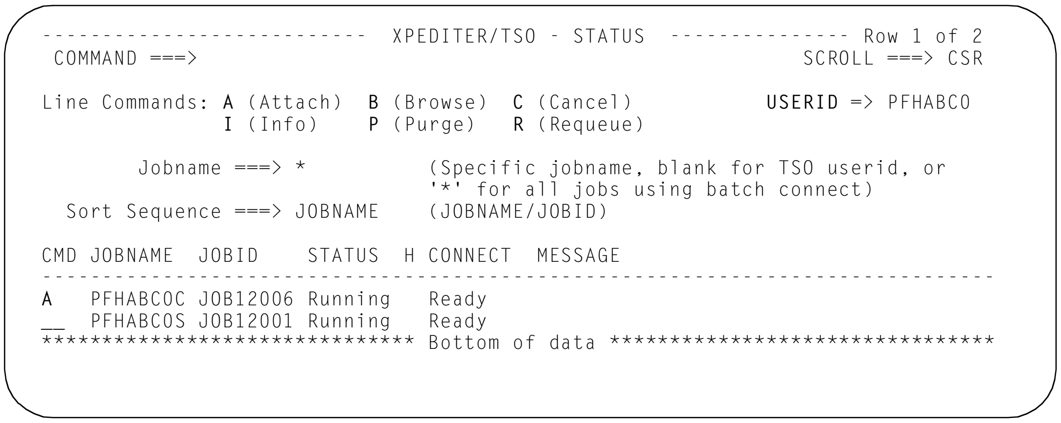
9. Using XPEDITER Batch Connect (which is OPTION 3 (BATCH) on the XPEDITER/TSO Primary Menu), set up the *Client application* that has been selected to call the Stored Procedure. It will duplicate steps 1 through 7.
10. Type an asterisk **(*)** in the "***Jobname***" field of the XPEDITER/TSO Status Screen # 2 (Figure 8-13) and press Enter to refresh the Status screen and view the status of both the *Client application* Job and the Stored Procedure.



```
---------------------------  XPEDITER/TSO - STATUS  --------------- Row 1 of 2
 COMMAND ===>                                                    SCROLL ===> CSR

Line Commands: A (Attach)  B (Browse)  C (Cancel)           USERID => PFHABCO
               I (Info)    P (Purge)   R (Requeue)

       Jobname ===> *          (Specific jobname, blank for TSO userid, or
                               '*' for all jobs using batch connect)
  Sort Sequence ===> JOBNAME   (JOBNAME/JOBID)

CMD JOBNAME  JOBID    STATUS  H CONNECT  MESSAGE
-------------------------------------------------------------------------------
__  PFHABCOC JOB12006 Running   Ready
__  PFHABCOS JOB12001 Running   Ready
****************************** Bottom of data *******************************
```

Figure 8-13. XPEDITER/TSO Status Screen # 2 (Client Application and Stored Procedure)

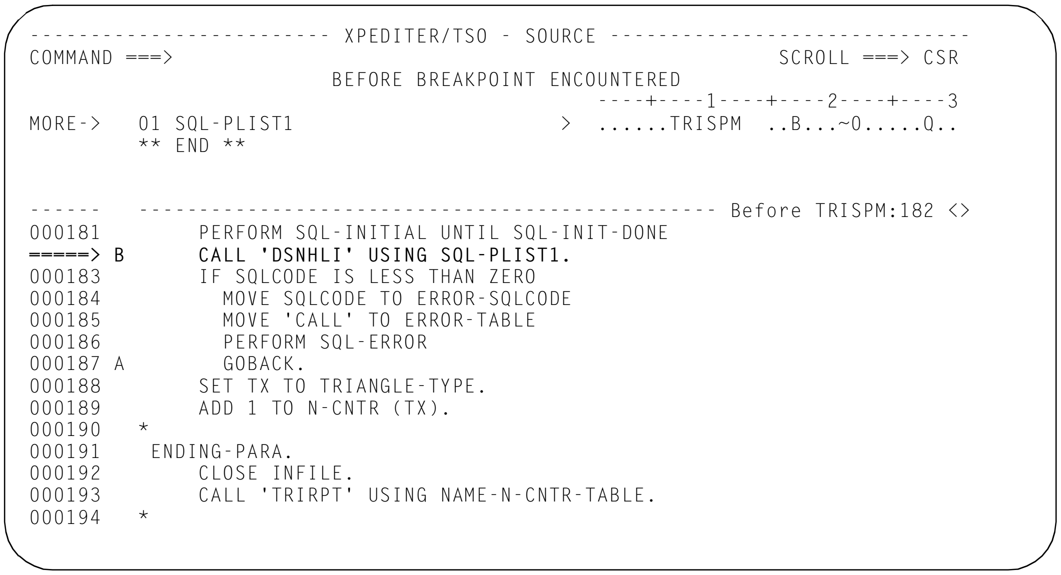
11. A new line of data is now displayed containing both a valid Jobname (your Client Jobname in the "***JOBNAME***" field) and a valid JOBID number. The "***CONNECT***" field should indicate that both the Client JOB and the Stored Procedure are *Ready* to connect. After you have verified that your Client JOB is *Ready* to connect, type an **A** (next to the Client Application JOBNAME) in the "***CMD***" field of the XPEDITER/TSO Status screen # 3 (Figure 8-14). Press Enter to connect to the *Client Application* JOB.



```
---------------------------  XPEDITER/TSO - STATUS  --------------- Row 1 of 2
 COMMAND ===>                                                SCROLL ===> CSR

Line Commands: A (Attach)  B (Browse)  C (Cancel)          USERID => PFHABCO
               I (Info)    P (Purge)   R (Requeue)

        Jobname ===> *          (Specific jobname, blank for TSO userid, or
                                '*' for all jobs using batch connect)
  Sort Sequence ===> JOBNAME    (JOBNAME/JOBID)

CMD JOBNAME  JOBID    STATUS  H CONNECT  MESSAGE
-------------------------------------------------------------------------------
A   PFHABCOC JOB12006 Running   Ready
__  PFHABCOS JOB12001 Running   Ready
****************************** Bottom of data *******************************
```

Figure 8-14. XPEDITER/TSO Status Screen # 3 (Client Application and Stored Procedure)

12. The source for the *Client application* should appear on the XPEDITER/TSO Source screen # 1 (refer to Figure 8-15). Enter a *Before* breakpoint on the line where the Stored Procedure is called.



```
------------------------ XPEDITER/TSO - SOURCE ------------------------------
COMMAND ===>                                                SCROLL ===> CSR
                         BEFORE BREAKPOINT ENCOUNTERED
                                               ----+----1----+----2----+----3
MORE->    01 SQL-PLIST1                     >  ......TRISPM  ..B...~0.....Q..
          ** END **


------    --------------------------------------------- Before TRISPM:182 <>
000181         PERFORM SQL-INITIAL UNTIL SQL-INIT-DONE
=====> B       CALL 'DSNHLI' USING SQL-PLIST1.
000183         IF SQLCODE IS LESS THAN ZERO
000184           MOVE SQLCODE TO ERROR-SQLCODE
000185           MOVE 'CALL' TO ERROR-TABLE
000186           PERFORM SQL-ERROR
000187 A         GOBACK.
000188         SET TX TO TRIANGLE-TYPE.
000189         ADD 1 TO N-CNTR (TX).
000190    *
000191     ENDING-PARA.
000192         CLOSE INFILE.
000193         CALL 'TRIRPT' USING NAME-N-CNTR-TABLE.
000194    *
```

Figure 8-15. XPEDITER/TSO Source Screen # 1 (Client Application)

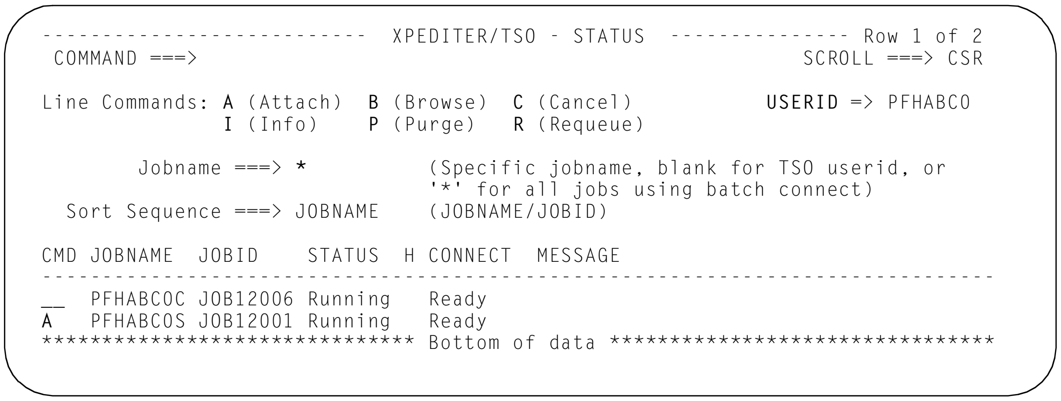
13. Enter any desired XPEDITER/TSO testing commands that you require.

14. When the *Before* breakpoint on the Stored Procedure call is encountered, type **GO 1** on the Command line and press Enter.

15. Immediately following issuance of the **GO 1** command, the *Client application* will '***lock***' (control has been transferred to the Stored Procedure). Press the Attention key twice. The screen that is displayed contains the following two lines:

```
XPED
ENTER ATTENTION OPTION OR HELP FOR LIST OF OPTIONS
```

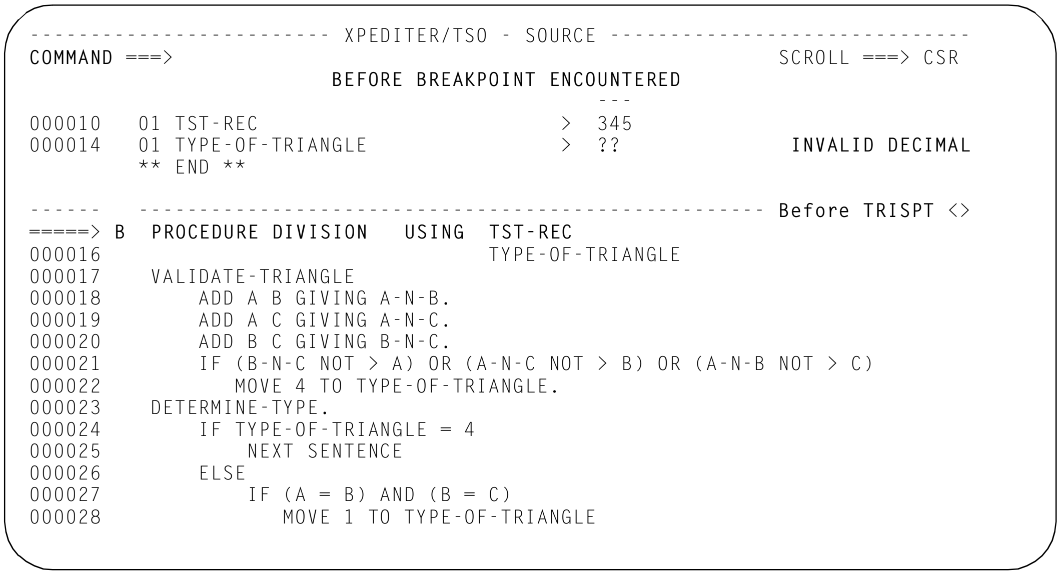
16. Type **DISC** (to disconnect) on the line below the message displayed on the screen, and press Enter. This will return you the Status screen without terminating the Client Application test.
17. Type an **A** (next to the Stored Procedure JOBNAME) in the "***CMD***" field of the XPEDITER/TSO Status screen # 4 (refer to Figure 8-16) and press Enter. This will connect you to the Stored Procedure.



```
---------------------------  XPEDITER/TSO - STATUS  --------------- Row 1 of 2
 COMMAND ===>                                                    SCROLL ===> CSR

Line Commands: A (Attach)  B (Browse)  C (Cancel)          USERID => PFHABCO
               I (Info)    P (Purge)   R (Requeue)

        Jobname ===> *          (Specific jobname, blank for TSO userid, or
                                '*' for all jobs using batch connect)
  Sort Sequence ===> JOBNAME    (JOBNAME/JOBID)

CMD JOBNAME  JOBID    STATUS  H CONNECT  MESSAGE
--------------------------------------------------------------------------------
__  PFHABCOC JOB12006 Running   Ready
A   PFHABCOS JOB12001 Running   Ready
****************************** Bottom of data ******************************
```

Figure 8-16. XPEDITER/TSO Status Screen # 4 (Client Application and Stored Procedure)

18. The source for the *Stored Procedure* should appear on the XPEDITER/TSO Source Screen # 2 (Figure 8-17).



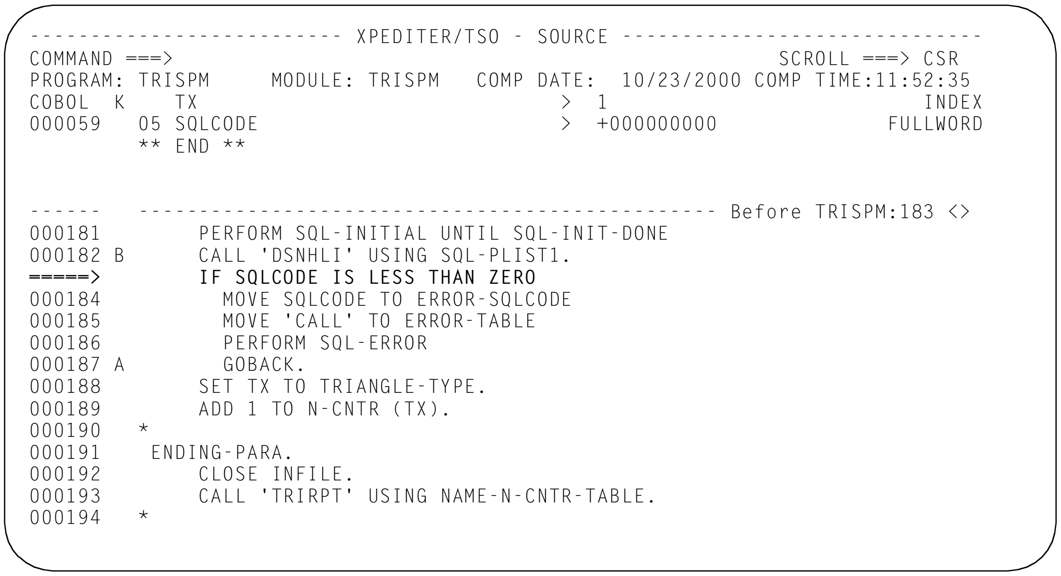
```
------------------------- XPEDITER/TSO - SOURCE -------------------------------
COMMAND ===>                                                    SCROLL ===> CSR
                         BEFORE BREAKPOINT ENCOUNTERED
                                                 ---
000010   01 TST-REC                         >  345
000014   01 TYPE-OF-TRIANGLE                >  ??               INVALID DECIMAL
         ** END **

------   --------------------------------------------------- Before TRISPT <>
=====> B  PROCEDURE DIVISION   USING  TST-REC
000016                               TYPE-OF-TRIANGLE
000017    VALIDATE-TRIANGLE
000018        ADD A B GIVING A-N-B.
000019        ADD A C GIVING A-N-C.
000020        ADD B C GIVING B-N-C.
000021        IF (B-N-C NOT > A) OR (A-N-C NOT > B) OR (A-N-B NOT > C)
000022           MOVE 4 TO TYPE-OF-TRIANGLE.
000023    DETERMINE-TYPE.
000024        IF TYPE-OF-TRIANGLE = 4
000025            NEXT SENTENCE
000026        ELSE
000027            IF (A = B) AND (B = C)
000028               MOVE 1 TO TYPE-OF-TRIANGLE
```

Figure 8-17. XPEDITER/TSO Source Screen # 2 (Stored Procedure)

19. Enter any desired XPEDITER/TSO testing commands that you require.
20. To return to the Client application, type **GO** at the end of the Stored Procedure and press Enter. If you had specified the number **1** in the "***Maximum number of Tests***" field when setting up the Stored Procedure for debugging with XPEDITER/TSO, you will now be placed in the LOG dataset with a **TEST COMPLETED** message. Entering **GO** will cleanly end the Stored Procedure test. If you had specified a number greater than **1** in the "***Maximum number of Tests***" field when setting up the Stored Procedure for debugging with XPEDITER/TSO, the Stored Procedure will '**lock**' (control has been returned to the Client application). Press the Attention key twice. The screen that is displayed contains the following two lines:

```
XPED
ENTER ATTENTION OPTION OR HELP FOR LIST OF OPTIONS
```

21. Type **DISC** (to disconnect) on the line below the message displayed on the screen, and press Enter. This will return you the Status screen without terminating the Stored Procedure Environment.
22. Type an **A** (next to the Client Application JOBNAME) in the "***CMD***" field of the XPEDITER/TSO Status screen # 3 (refer to Figure 8-14 on page 8-14) and press Enter. This will connect you to the *Client Application*.
23. You should now be able to view the XPEDITER/TSO Source Screen # 3 (refer to Figure 8-18) that displays the *Client Application*, just after the Stored Procedure call.



```
------------------------- XPEDITER/TSO - SOURCE -----------------------------
COMMAND ===>                                                  SCROLL ===> CSR
PROGRAM: TRISPM     MODULE: TRISPM   COMP DATE:  10/23/2000 COMP TIME:11:52:35
COBOL  K    TX                           >  1                           INDEX
000059   05 SQLCODE                      >  +000000000               FULLWORD
         ** END **


------   --------------------------------------------------- Before TRISPM:183 <>
000181        PERFORM SQL-INITIAL UNTIL SQL-INIT-DONE
000182 B      CALL 'DSNHLI' USING SQL-PLIST1.
=====>        IF SQLCODE IS LESS THAN ZERO
000184          MOVE SQLCODE TO ERROR-SQLCODE
000185          MOVE 'CALL' TO ERROR-TABLE
000186          PERFORM SQL-ERROR
000187 A        GOBACK.
000188        SET TX TO TRIANGLE-TYPE.
000189        ADD 1 TO N-CNTR (TX).
000190   *
000191    ENDING-PARA.
000192        CLOSE INFILE.
000193        CALL 'TRIRPT' USING NAME-N-CNTR-TABLE.
000194   *
```

Figure 8-18. XPEDITER/TSO Source Screen # 3 (Client Application)

24. Continue repeating steps 1 through 23 to switch back and forth between the *Stored Procedure* and the *Client application.* For the Stored Procedure, when the "***Maximum number of Tests***" (specified when you first set up the Stored Procedure test) has been reached, you will be positioned in the XPEDITER LOG dataset screen with the message: **TEST COMPLETED**. When this message appears, type **GO** on the command line and press Enter to exit the Stored Procedure.

DB2 Stored Procedure Security Considerations

Be advised that there are a few security considerations concerning XPEDITER/TSO Stored Procedure support.

Batch Connect Security Check

Connection can be made to any job, including production jobs, if you permit testing in production DB2 subsystems, as long as your site security grants the authority. The batch connect facility is shipped with a default security exit routine that allows connection to a job if the JOBNAME, without the last character, matches the TSO ID where the STATUS panel is accessed. When a connection cannot be made, the messages `CANNOT CONNECT` or `SECURITY CHECK FAILED` are issued.

The site installer can customize the security exit routine to tailor the security level for certain groups or individuals. When an asterisk (*) is entered in the "***Jobname***" field on the Status screen, all jobs that are waiting for connection or being tested under batch connect are listed. System programmers are able to connect to a remote job and use the facility as a help desk feature in debugging application programs.

DB2 Subsystems to Debug

Before XPEDITER/TSO can be used to debug Stored Procedures, a XPEDITER/TSO Stored Procedure activation program must be run on the OS/390 system (after every IPL). The DB2 subsystems XPEDITER/TSO will be allowed to intercept are specified to the activation program. Refer to the *XPEDITER/TSO and XPEDITER/IMS Installation Guide* for details on the execution of the DB2 Stored Procedure activation program.

Optional Security Access Facility (SAF) Calls

Optionally, XPEDITER/TSO will issue SAF (Security Access Facility) calls to your external security package (RACF, ACF2, Top Secret,...etc.). This allows your installation to limit the number of individual users who are able to debug DB2 Stored Procedures being called from whichever client selected. Refer to the *XPEDITER/TSO and XPEDITER/IMS Installation Guide* for details on how to add additional DB2 Stored Procedure security.

Security Exit

If you have specific security issues not addressed by the aforementioned security procedures, you can optionally code an XPEDITER/TSO and XPEDITER/IMS security exit. Refer to the *XPEDITER/TSO and XPEDITER/IMS Installation Guide* for details on how to add additional DB2 Stored Procedure security via the ADSRAUSR exit.

Chapter 9. Debugging Programs With Special Conditions

ESA Support

XPEDITER/TSO runs as an ISPF dialog in the TSO region under the MVS/ESA operating system. XPEDITER/TSO modules reside above the 16 MB line, except for the I/O modules. You can execute any programs that are AMODE(31) or RMODE(ANY) with XPEDITER/TSO.

Code Generator Support

Source code that has been generated with a code generator such as INTERSOLV APS, IBM Cross System Product Application Development (CSP/AD), and CA-Telon, can be compiled with the language processor and debugged with XPEDITER/TSO at the generated COBOL level. Code generators that produce object modules are executed with XPEDITER/TSO; however, you might not be able to do any symbolic debugging, such as setting breakpoints and displaying variable contents.

Optimized Code Support

COBOL programs that are optimized using the CA-OPTIMIZER or the OPTIMIZE compiler option can be debugged symbolically under XPEDITER/TSO without altering the load module. However, depending on the optimizing algorithm applied to the code, execution trace and stepping through code in XPEDITER/TSO can appear to be incorrect. The logic optimization can also impact the data flow in the program. Data values displayed with the KEEP and PEEK commands cannot be updated correctly, and the GOTO command is not allowed for a procedure name or statement number that was streamlined by the optimizer.

The side effects of optimization can be verified by comparing the test results against code that has not been optimized, or by analyzing the PMAP and LIST information generated at compile time. When processing the source listing member with the VS COBOL II compiler, the LIST option is required with the OPT option, although the OFFSET option is accepted with the NOOPT option. The CA-OPTIMIZER II is not supported by XPEDITER/TSO. However, the CA-OPTIMIZER III Releases 5.1 and 6.0 **are** supported by XPEDITER/TSO.

VS COBOL II Releases 3.0 and Above (COBOL 85) Support

COBOL programs compiled with VS COBOL II Release 3.1 (3.0) can be processed by the language processor without the CMPR2 option, and can be debugged with XPEDITER/TSO. The ANSI 85 features, such as mixed lowercase and uppercase, use of external data and files, reference modifications, in-line performs, and scope terminators, are supported. However, nested programs and omission of the Procedure Division are not supported by XPEDITER/TSO.

Also, OS/VS COBOL and VS COBOL II Release 2.0 compiled programs executed with the VS COBOL II Release 3.1 (3.0) run-time library can be debugged under XPEDITER/TSO. Programs using MIXRES and IGZBRIDGE options are also supported.

Mixed Language Support

You can debug mixed COBOL, PL/I, and/or Assembly language applications under the same debugging session if you are licensed for the languages comprising the application. If you are not licensed, you cannot create the necessary source listing records.

If you are licensed, XPEDITER/TSO automatically enables the appropriate language commands and run-time environment for the qualified program. The source is displayed if you created the source listing member for the program; the language dependencies are transparent to the user. Otherwise, the module for the language that is not licensed is treated as an executable module without source. You are not allowed to perform any symbolic debugging tasks; however, you can execute through the module.

Subroutine Testing Support

You can debug COBOL subroutines in the unit testing mode or in the integration testing mode. When debugging a COBOL subroutine stand-alone, specify the subroutine name on the test screen. If the subroutine name differs from the load module name, also specify the load module name.

When the source is displayed, you must initialize the Linkage Section by moving in parameters, as if the call had just been made by the driver program. You can facilitate this process and create the source listing member only for the subroutine that you are debugging. Specify the driver program name on the test screen even though there is no source listing available for the driver.

When the debugging session starts, the log file is displayed, however, the execution status indicates that the program is at the beginning of the driver. Enter a before module breakpoint on the submodule, then resume execution. XPEDITER/TSO executes the driver, the driver calls the submodule, and the source is displayed as soon as the breakpoint is encountered.

The inserted ACCEPT command can be very useful when you want to initialize Linkage Section values. For information on the ACCEPT command, refer to the *XPEDITER/TSO and XPEDITER/IMS Reference Manual*.

Database Support

There are three types of databases that XPEDITER/TSO supports. The first category is third-party software in general, the second is IMS/DB, and the last is DB2.

IDMS/DB, ADABAS, SUPRA, DATACOM/DB, System 2000

You can debug batch programs that issue calls to IDMS/DB, ADABAS, SUPRA (TIS/TOTAL), DATACOM/DB, and System 2000 under the Standard environment (option 1 on the XPEDITER/TSO Environments Menu). The databases are treated as if the programs are issuing I/O operations to the VSAM files. You must pre-allocate the databases using the File Allocation Utility. There are no special parameters that you must set in order to debug ADABAS and SUPRA programs; however, you must SET STATIC OFF to access IDMS database programs. Refer to the *XPEDITER/TSO and XPEDITER/IMS Reference Manual* for information about the SET commands.

IMS/DB

Batch programs that make database calls using CALL to CBLTDLI are debugged in the IMS/DB environment (option 3 on the XPEDITER/TSO Environments Menu). Verify that the IMS-related libraries, such as RESLIB, PSB, and DBD, and the parameters information are correctly defined in the Setup Options screens as well as on the IMS test screen.

During the debugging session, you can dynamically issue DLI function calls on behalf of your program and access the IMS databases through the use of the DLI command.

IOPCB and return codes can be kept in the Keep window to monitor the DLI calls that are issued.

The last function code and its return code can be retrieved to monitor the DLI activities through the use of the SHOW IMSFUNC command. Using HELP IMSCODE shows the common IMS abend codes.

DBT HSSR

DBT HSSR is the abbreviation for the IBM IMS/VS Data Base Tool High Speed Sequential Retrieval utility. XPEDITER can be used to test both DBT HSSR application programs and user exits.

These programs are debugged using the IMS/DB environment option (option 3 on the XPEDITER/TSO Environments Menu). To support HSSR, you must add one or more PARMs on the XPEDITER/TSO IMS Parameter Lists screen. For example:

```
TYPE     PARM LIST

HSD      DFSRRC00/DLI,MODULE,PSB,...
HSB      DFSRRC00/DBB,MODULE,PSB,...
HSU      DFSRRC00/ULU,MODULE,DBD,...
```

The PARM must follow HSSR standards. To get the proper parameter values, refer to the working batch JCL or to the *IBM IMS/VS DBT HSSR User's Guide*.

Begin the PARM with the name of the control program, normally DFSRRC00, followed by a slash (/), followed by the environment type.

The subparameters following the first and second commas (that is, MODULE and either PSB or DBD) are dummy values that are replaced at execution time by values from the IMS test screen.

On the IMS test screen, specify the program type that matches the HSSR PARM you want; i.e., HSD, HSB, or HSU.

In the PROGRAM field on the IMS test screen, specify the highest level HSSR application program. This is the program name that is normally specified as the MBR keyword in the JCL PARM. For testing HSSR user exits, the name is either FABHFSU or FABHURG1.

XPEDITER uses the value in the PSB field internally to replace the data string between the second and third commas. This can be either a PSB or a DBD, depending on the HSSR function to be performed.

Since the XPEDITER/TSO IMS Setup Menu has screens only for IMS-related datasets, allocate the HSSR datasets in your file allocation list.

When you press the **Enter** key from the XPEDITER/TSO IMS test screen, the presence of the slash (/) in the specified PARM causes XPEDITER to automatically invoke the HSSR driver program, FABHX034. Before and after module breakpoints are set automatically in the program specified on the test screen. If no source listing member exists for this program, the XPEDITER log is displayed. From the log, you can set additional breakpoints and then enter the GO command.

DB2

Batch programs that make EXEC SQL calls without IMS/DB are debugged in the Standard environment (option 1 on the XPEDITER/TSO Environments Menu). The DSNLOAD libraries and the associated DB2 subsystem name must be identified on the Setup screen. Enter **YES** in the Is this a DB2 Test? field. In this case, XPEDITER/TSO issues a DSN RUN command to establish the DB2 environment. The plan name and the system name also need to be filled out correctly to successfully execute the DSN RUN.

When the source is displayed, you will notice that XPEDITER/TSO defaults to suppressing the DB2-translated statements, except for the initialization statements. The EXEC SQL statements can be optionally expanded to be debugged at the generated COBOL statement level instead of at the EXEC SQL level by using the GEN command. Also, if you are licensed for XPEDITER for DB2 Extension and File-AID for DB2, you can insert EXEC SQL statements to prototype DB2 calls in addition to browsing and editing DB2 tables while in XPEDITER/TSO.

ISPF dialog applications that have programs that make EXEC SQL calls are debugged in the Dialog environment (option 2).

Batch programs that make DB2 calls as well as IMS/DB calls must be debugged in the IMS/DB environment (option 3).

IMS/DC programs that make DB2 calls are debugged with BTS in the BTS environment (option 4) or by using XPEDITER/IMS option 8 or 9. In an IMS/DC test session with DB2, split screen cannot be used with any product that causes another attach to DB2. This is an IMS restriction.

Use HELP SQLCODE to show the common SQL return codes.

Shared DL/I Database (DFHDRP) Support

Programs that access DL/I databases shared between the CICS region and TSO can be debugged through XPEDITER/TSO by executing a CLIST through TSO or by executing a CLIST as a USEREXIT.

The XPEDITER/TSO support for DFHDRP is provided for users who have already installed DFHDRP (Shared DL/I Database) at their sites. See the *CICS/OS/VS Installation and Operations Guide*, order number SC33-0071, "Running a Batch Region for DL/I Shared Databases," for details of this IBM feature.

There are two methods of installing XPEDITER/TSO DFHDRP support:

1. USEREXIT CLIST	Invoked from within XPEDITER/TSO and is panel driven.
2. CLIST to execute DFHDRP	Invoked through TSO. Requires modification for each program to be tested under XPEDITER/TSO.

To install this support, refer to "Appendix C. Shared DL/I Database (DFHDRP) Support" in the *XPEDITER/TSO and XPEDITER/IMS Installation Guide*.

After the CLIST is executed, the program source appears. At this point, normal XPEDITER/TSO commands are in effect.

Note: The RETEST command does not work in this environment since the program is treated as an IMS/VS batch program by XPEDITER/TSO.

SORT EXIT Support

XPEDITER/TSO can debug sort exits for DFSORT and SYNCSORT if you follow these steps:

1. Enter **XPDSORT** as the program name on the test screen instead of your sort exit program name. XPDSORT is a dummy program supplied by XPEDITER/TSO to create a link to SORT. You must specify the name of the XPEDITER load library on the Load Module Libraries screen. This library contains XPDSORT. The Load Module Libraries screen can be accessed from the Test Setup Menu selected from the Standard test environment screen.
2. Enter the following, plus any parameters required by your specific sort, in the PARM string field of the test screen:

   ```
   NOSTIMER
   ```

   For DFSORT, specify:

   ```
   DEBUG NOESTAE
   ```

   using the DFSPARM, SYSIN, or SORTCNTL DD statement.
3. Create an initial test script member that contains the following commands:
   - **SET DYNAMIC eeeeeeee**

     where *eeeeeeee* is the name of your sort exit.

     Notes:

     a. Ensure that the specified sort exit exists in the user load library under SETUP.

     b. You must also modify your sort control statement, such as MODS Enn=(eeeeeeee,10,,C), to point to XTASKLIB.
   - **SET DYNAMIC xxxxxxxx**

     where *xxxxxxxx* is the name of any dynamically called load module to be debugged.

     Note: Ensure that the specified load module exists in the user load library under SETUP.
   - **BEFORE xxxxxxxx:**

     where *xxxxxxxx* is the name of any module to be debugged, compiled by JCL calling XPEDITER/TSO.

     Note: Remember to place a colon (:) at the end to specify that this is a compiled module.
   - **AFTER xxxxxxxx:**

     where *xxxxxxxx* is the name of any module to be debugged, compiled by JCL calling XPEDITER/TSO.

     Note: Remember to place a colon (:) at the end to specify that this is a compiled module.
   - **SET EXCLUDE zzzzzzzz**

     where *zzzzzzzz* is the name of any COBOL module that does VCON modifications.

   Note: If the sort exit being tested has a load module name different than the CSECT name, use the fully qualified BEFORE and AFTER commands. For example, use the following format:

   ```
   BEFORE load module::SORT_EXIT:
   ```

4. Verify that the file allocation list specified on the test screen contains all of your sort work files and sort control statements, either in-stream or in a dataset allocated under one of the following DD statements—SYSIN, DFSPARM, SORTCNTL, or $ORT-PARM (for SYNCSORT), in addition to the files your application programs are expecting.

Note: If a debugging session must be terminated prematurely while SORT is active, do not use the EXIT command (it can cause a S0C4 abend), but use the following primary commands instead:

```
MOVE 16 TO SORT-RETURN
GO 1
GOTO <RELEASE statement number>  or  GOTO <RETURN statement number>
      (depending on whether the SORT is active in the INPUT procedure or
       the OUTPUT procedure)
GO 1  (Enter GO 1 until you are out of the INPUT or OUTPUT procedure,
      depending on where the SORT is active.  You should be at the
      statement after the SORT.)
GOBACK
```

CAUTION:
Repetitive premature exits from SORT can diminish the region size.

PANEXEC Support

PANEXEC programs can be debugged using XPEDITER/TSO. The installer must enter **YES** in the PANEXEC Load Module Support Installed field on the Installation Options screen. This allows the PANEXEC Load Libraries screen to appear during installation.

On the PANEXEC Load Libraries screen, the dataset name allocated to ddname PANESRL must be entered in the PANEXEC Load Library DSNAMES field and the ddname associated with your PANEXEC control cards must be entered in the PANEXEC Control Card File DSNAMES field.

COBOL/370 and LE/370 Support

XPEDITER/TSO provides the following support for COBOL/370 and LE/370:

- Debugging applications with TRAP ON or TRAP OFF. Applications can be debugged with TRAP ON as well as TRAP OFF. The SET LETRAP ON/OFF command lets you set the TRAP option.
- Debugging the new handlers language construct. XPEDITER/TSO lets you control execution and follow user handler invocation with the following restriction:
  - **User handler for S0C1**: XPEDITER/TSO will always get control of a S0C1 abend, preventing the user from effectively debugging handler routines that process S0C1 abends.

Usage Note

Even if your application programs would normally find any required Language dependent run-time subroutines (including LE - Language Environment), without being included in the JOBLIB/STEPLIB of the batch JCL (usually from the LINKLIST or (E)LPA), the libraries must still be specified as part of the test session setup. This will ensure that XPEDITER's Task Library will be properly configured. For Batch Connect, the preferred method is to include the run-time libraries in the STEPLIB DD statement concatenations of the JCL step(s) that are being intercepted.

ADCYCLE PL/I and LE/370 Support

XPEDITER/TSO provides the following support for ADCYCLE PL/I and LE/370:

- Debugging applications with TRAP ON or TRAP OFF. Applications can be debugged with TRAP ON as well as TRAP OFF. The SET LETRAP ON/OFF command lets you set the TRAP option.
- Debugging the new handlers language construct. XPEDITER/TSO lets you control execution and follow user handler invocation with the following restriction:
  - **User handler for S0C1**: XPEDITER/TSO will always get control of a S0C1 abend, preventing the user from effectively debugging handler routines that process S0C1 abends.

    Note: When using ASSEMBLER, initial breakpoints or the setting of breakpoints on a CEEENTRY macro are not allowed. XPEDITER/TSO cannot permit breakpoints on this macro because of conflicts with LE/370 abend processing.

Usage Note

Even if your application programs would normally find any required Language dependent run-time subroutines (including LE - Language Environment), without being included in the JOBLIB/STEPLIB of the batch JCL (usually from the LINKLIST or (E)LPA), the libraries must still be specified as part of the test session setup. This will ensure that XPEDITER's Task Library will be properly configured. For Batch Connect, the preferred method is to include the run-time libraries in the STEPLIB DD statement concatenations of the JCL step(s) that are being intercepted.

Third-Party Package Support

Third-Party packages are often written in Assembler language using a non-standard linkage convention. Reset the static call tracing facility by using the SET STATIC OFF command in an initial script so that XPEDITER/TSO does not monitor the calls.

Global Handling Of Special Conditions

If special conditions can be handled by a command stream in an initial script, the @@SITE@@ member can be used to include these commands for the entire user community. This member will be executed prior to any initial script specified by the user. If placed in a Site-wide Script Library (defined at installation), a special environment can be initiated for all users, even if no initial script is specified.

Using MQSeries With XPEDITER/TSO

In order to use MQSeries with XPEDITER/TSO, you must allocate MQSeries LOADLIBs under DFSESL.

There are two methods of allocating MQSeries LOADLIBs under DFSESL:

1. By indicating **DB2 = Y** with a subsystem name (on the test screen). The subsystem name points to an entry on the DSN LOADLIB screen in the **SEtup**.

   On the DSN LOADLIB screen, more than one LOADLIB can be indicated under *DSN LOAD DSNAME* for any given ***NAME***.

   For example, when you are using DB2:

   ```
   NAME      DSNLOAD DSNAME
   D220      'Valid.DB2.Subsystem'
   D220      'MQ.SERIES.LOADLIB'
   ```

   In another example, when you are **NOT** using DB2:

   Answer **DB2 = Y** in every instance.

   ```
   NAME      DSNLOAD DSNAME
   D220      'MQ.SERIES.LOADLIB'
   ```

2. Use the File Allocation Utility (FAU). Refer to Appendix H, page H-18 of the *XPEDITER/TSO and XPEDITER/IMS Installation Guide* for information about DSNLOAD libraries and datasets.

Appendix A. Using the File Allocation Utility

The File Allocation Utility (FAU) is a file list editor that is used mainly to create and maintain file allocation lists for later use. When problems occur during file allocations, the FAU is automatically accessed and the FAU screens required to correct the problem are displayed. The screens and procedures available with the FAU are described in this section.

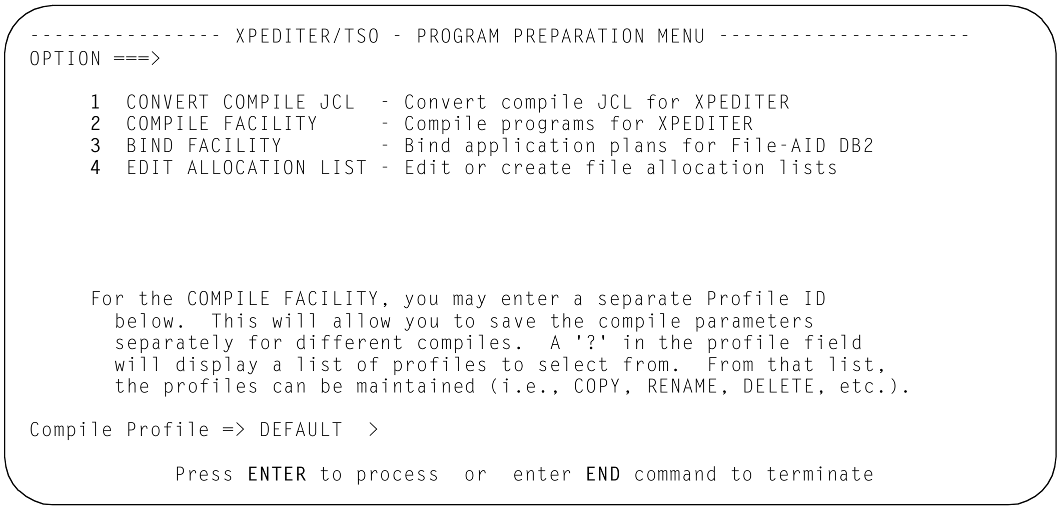
Using the FAU, you can:

- Create new file lists.
- Edit existing file lists.
- Copy existing JCL, CLIST, and file lists and automatically select the DD statements to be copied to a new file list.
- Save and allocate a file list for immediate use, or save the file list and allocate it at another time.
- Depending on your site specifications, access JCL from some library management systems, such as Librarian, Panvalet, Endeavor, and so on.
- Allocate any number of files at one time, up to the limit defined by your shop.

Accessing the File Allocation Utility

Use option 1 (Prepare) on the XPEDITER/TSO Primary Menu to access the FAU. The Program Preparation Menu shown in Figure A-1 is displayed.

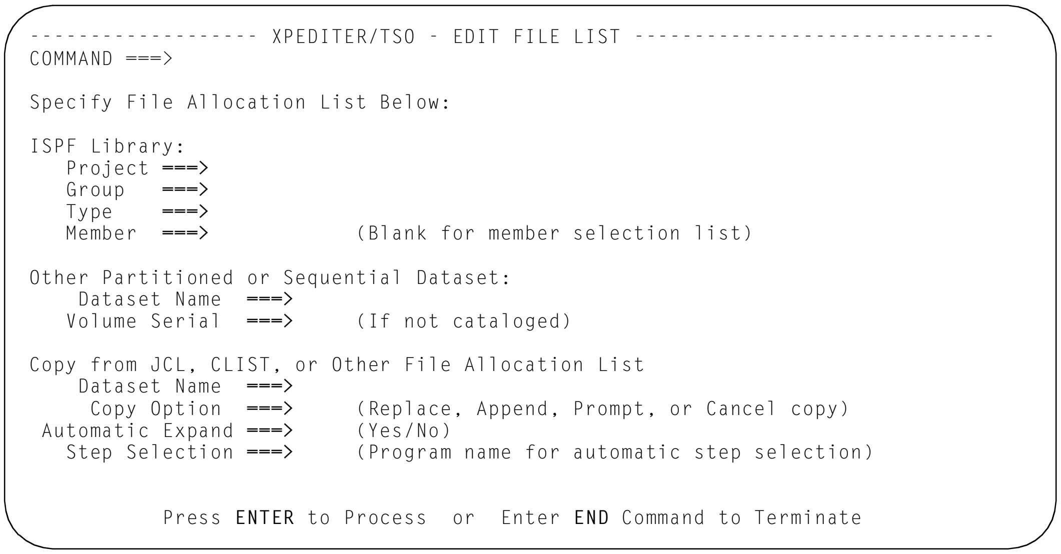
Note: The FAU is automatically accessed when problems occur during file allocations from the test screen or using the ALLOCATE command. Only the screens needed to correct the problem are displayed.



```
---------------- XPEDITER/TSO - PROGRAM PREPARATION MENU ---------------------
OPTION ===>

     1  CONVERT COMPILE JCL  - Convert compile JCL for XPEDITER
     2  COMPILE FACILITY     - Compile programs for XPEDITER
     3  BIND FACILITY        - Bind application plans for File-AID DB2
     4  EDIT ALLOCATION LIST - Edit or create file allocation lists





     For the COMPILE FACILITY, you may enter a separate Profile ID
       below.  This will allow you to save the compile parameters
       separately for different compiles.  A '?' in the profile field
       will display a list of profiles to select from.  From that list,
       the profiles can be maintained (i.e., COPY, RENAME, DELETE, etc.).

Compile Profile => DEFAULT  >

            Press ENTER to process  or  enter END command to terminate
```

Figure A-1. Program Preparation Menu

Use option **4** (Edit Allocation List) to access the FAU. The Edit File List screen shown in Figure A-2 on page A-2 is displayed.



```
------------------ XPEDITER/TSO - EDIT FILE LIST ------------------------------
COMMAND ===>

Specify File Allocation List Below:

ISPF Library:
   Project ===>
   Group   ===>
   Type    ===>
   Member  ===>            (Blank for member selection list)

Other Partitioned or Sequential Dataset:
    Dataset Name  ===>
   Volume Serial  ===>     (If not cataloged)

Copy from JCL, CLIST, or Other File Allocation List
    Dataset Name  ===>
     Copy Option  ===>     (Replace, Append, Prompt, or Cancel copy)
 Automatic Expand ===>     (Yes/No)
   Step Selection ===>     (Program name for automatic step selection)


        Press ENTER to Process  or  Enter END Command to Terminate
```

Figure A-2. Edit File List Screen

The dataset name of the file allocation list you are creating, editing, or copying is entered in either the top (ISPF Library) or middle (Other Partitioned.....) section of the Edit File List screen.

Notes:

1. If you enter the name of an existing dataset that **does not** contain a file list, an error message is displayed.
2. If you enter the name of a PDS in any of the fields on this screen and you do not enter a member name, a member selection list is displayed. You can select the member from the list.

The way you complete the screen indicates what you want to do and determines the sequence in which the FAU screens are displayed.

Creating a New File List

Enter a new, fully qualified dataset name in either the top (ISPF Library) or middle (Other Partitioned.....) section of the screen. You are presented with the first of the Edit File List screens. These screens will not contain any values. You can manually enter the information about the files to be allocated. Refer to "Using the Edit File List Screens" on page A-4 for a description of how to use these screens.

Editing an Existing File List

Enter the dataset name of an existing file list in either the top (ISPF Library) or middle (Other Partitioned....) section of the screen. You are presented with the first of the Edit File List screens. The parameters associated with each file in the file list are displayed on these screens. The commands listed at the top of the screens can be used to browse and edit the data, copy data, allocate the file list, and so on. Refer to "Using the Edit File List Screens" on page A-4 for a description of how to use these screens.

Copying Existing File Lists, JCL, or CLISTs

Enter a new fully qualified dataset name or a new member name for an existing dataset in either the top (ISPF Library) or middle (Other Partitioned°) section of the screen. Enter the name of the dataset to be copied, the copy option, expand option, and step selection in the bottom section of the screen.

The following occurs when the copy option operates successfully:

- If the dataset being copied contains a file list, the parameters associated with each file are listed on the appropriate Edit File List screens. The commands listed at the top of the screens can be used to browse and edit the data, copy data, allocate the file list, and so on. Refer to "Using the Edit File List Screens" on page A-4 for a description of how to use these screens.
- If the dataset being copied contains JCL, the JCL is displayed on the Select DDNAME screen.

  You can select the DD statements you want copied to the file list and perform other functions, such as editing. Refer to "Converting JCL to a File List" on page A-12.

The Copy Option choices are described below:

Replace
The existing file list is replaced by the data being copied.

Append
The data being copied is appended to the end of the existing file list.

Prompt
The default. If the "copy to" dataset already exists and Prompt is entered, the Copy Option screen shown in Figure A-3 is displayed. This screen verifies the names of the existing file list and the source to be copied. Press **Enter** to continue or enter one of the options listed for the Copy Option field on the screen.

Cancel copy
The copy operation is suppressed, and you will enter the FAU with the existing file list displayed on the appropriate Edit File List screens.

END
If the END key is used, the editing session is ended and you are returned to the Edit File List data entry screen.

The remaining fields are:

Automatic Expand
This field is used if the dataset being copied executes JCL procedures. If set to Yes, the references to the JCL procedures will be expanded to the actual JCL that is executed. Refer to "Things to Know About JCL Expansion" on page A-14.

Step Selection
If you enter a program name in this field, and a step in the JCL executes that program, that step is automatically selected and the JCL is converted to the file list.

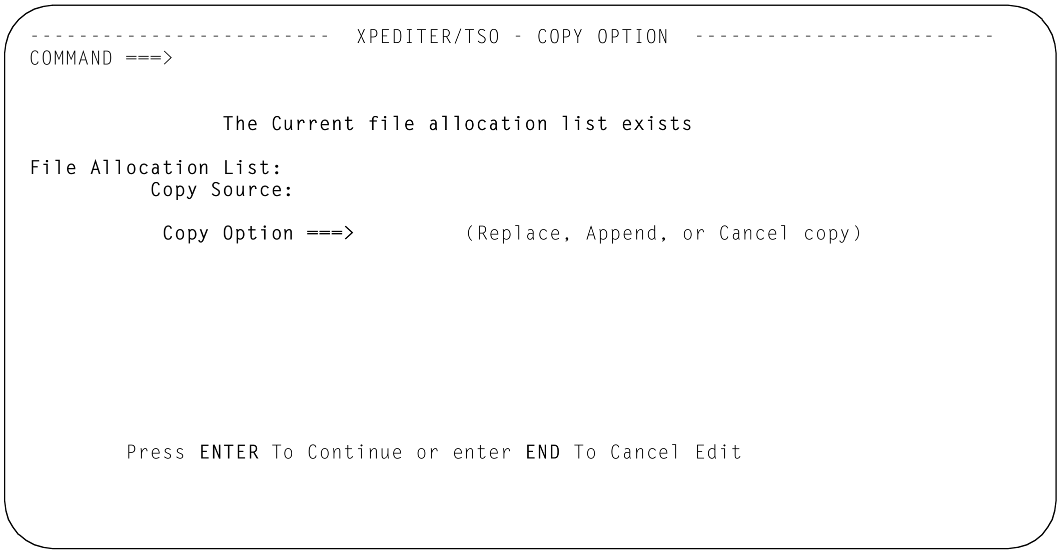

```
------------------------  XPEDITER/TSO - COPY OPTION  ------------------------
COMMAND ===>


                The Current file allocation list exists

File Allocation List:
          Copy Source:

           Copy Option ===>         (Replace, Append, or Cancel copy)









        Press ENTER To Continue or enter END To Cancel Edit
```

Figure A-3. Copy Option Screen

Saving a File List

After you enter, edit, or copy the necessary ddname information, you are returned to the Edit File List screen. From this screen, use either the SAVE or END command to save the file, or use the ALLOCATE command to allocate the files. If there are errors, the appropriate Edit File List screen is displayed with the ddname in question scrolled to the top of the screen. For a verification error, a message appears in the upper right corner of your screen; if the error is the result of the allocation process, a long error message appears centered on the third line.

Using the Edit File List Screens

The parameters associated with each file in the file list are specified on the Edit File List screens.

If your installation **does not** use SMS to manage DASD datasets, the screens shown in Figure A-4 are displayed.

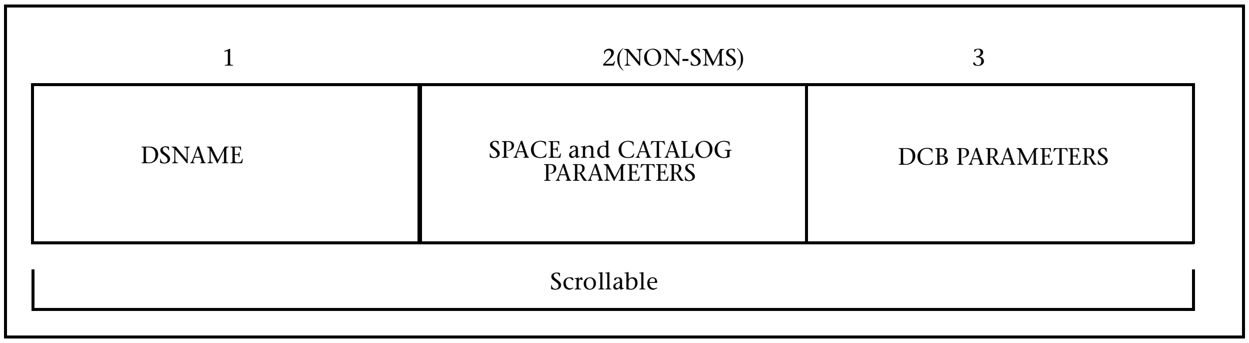


Figure A-4. Edit File List Screens Without SMS

If your installation uses SMS to manage DASD datasets, the screens shown in Figure A-5 are displayed.

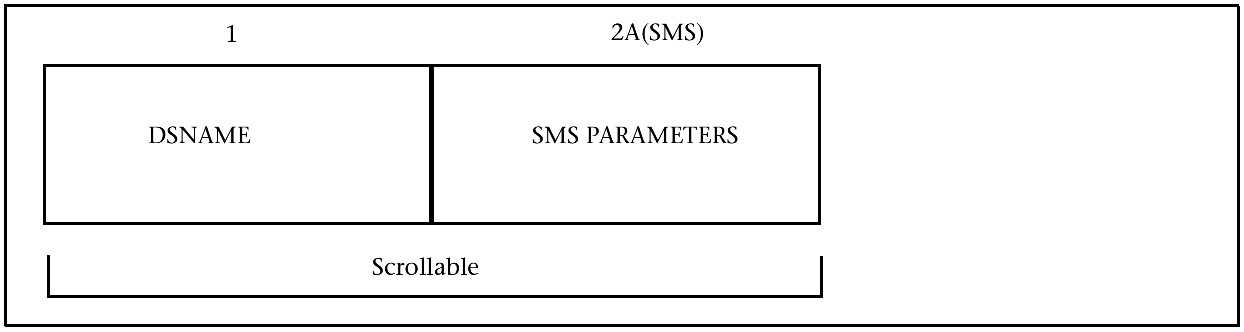


Figure A-5. Edit File List Screens With SMS

The RIGHT and LEFT scrolling commands let you move from one screen to another. The UP and DOWN scrolling commands are also available. Each screen is numbered, and contains information for the following categories:

1. **Edit File List 1 -** DDNAME, dsname, and disposition.
2. **Edit File List 2 -** Space and catalog parameters or NON-SMS parameters.

   Note: If your installation uses SMS to manage DASD datasets, the Edit File List 2 screen displays the data class, storage class, management class, and delete option.

2A. **Edit File List 2A** - SMS parameters

3. **Edit File List 3** - DCB parameters.

You can enter the parameters manually or type over the existing parameters to change the values.

The Edit File List screens show primary and line commands at the top of each screen. Note that the primary commands listed on each screen may vary, depending on the manner in which the FAU is accessed.

The line commands are entered by typing over the line command area (leftmost column of each screen). You can delete, insert, or repeat lines to make changes to the file list. The primary and line commands are described below:

Primary Commands

CANcel	Exits the FAU without allocating or saving the file list. You are returned to the previous screen.
COPY	Copies an existing file list, JCL, or CLIST. Refer to "Using the COPY Command on Edit File List Screens" on page A-7 for more information.
ALLOCate	Allocates the file list and returns you to the Edit File List screen.
SAVE	Saves the file list and returns you to the Edit File List screen.
END	Terminates the FAU. Depending on the way you entered the FAU, the END command also allocates and/or saves the file list.
HELP	Displays interactive HELP.

Line Commands

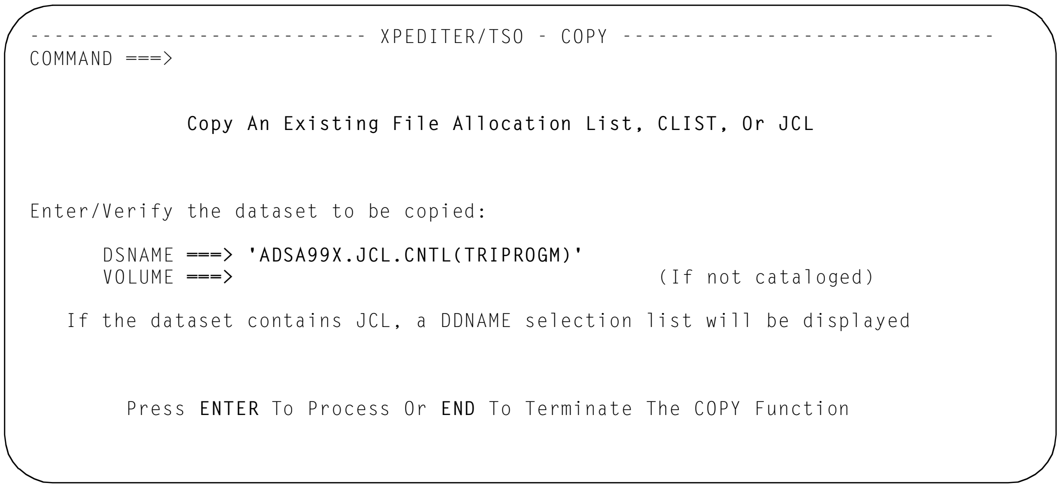
Dn (Delete)	Deletes a line, where *n* is the number of lines to delete. The ddname on the line and all related parameters are parameters are deleted. The block form (DD) of the command can be used to delete a block of lines.
In (Insert)	Inserts a blank line, where *n* is the number of lines to insert.
Rn (Repeat)	Repeats the line, where *n* is the number of times the line is to be repeated. The ddname and all related values are repeated. The block form (RR) can be used to repeat a block of lines.
S (Select detail)	Displays detail information about the DCB, dataset allocation, security, and SYSOUT parameters for the selected ddname. The detail information is displayed on the File PARMS screens described in "Displaying File Parameters" on page A-10.
BR (Browse dataset)	Displays the dataset referenced by the selected ddname.
ED (Edit dataset)	Displays the dataset referenced by the selected ddname for editing.

Using the COPY Command on Edit File List Screens

When you use the COPY command on any of the Edit File List screens and you do not enter a dataset name, the Copy screen shown in Figure A-6 on page A-7 is displayed for entering the dataset name. **Entering the COPY command with a dataset name skips this screen.**

In the DSNAME field, enter the name of the dataset to be copied. Enter a value for the VOLUME field only if the dataset is not cataloged. The dataset can be either sequential or partitioned, fixed or variable block, with a record length of 80 or greater.

Note: If the name of a PDS is entered without a member name, a member selection list is displayed.



```
---------------------------- XPEDITER/TSO - COPY -------------------------------
COMMAND ===>


             Copy An Existing File Allocation List, CLIST, Or JCL



Enter/Verify the dataset to be copied:

      DSNAME ===> 'ADSA99X.JCL.CNTL(TRIPROGM)'
      VOLUME ===>                                    (If not cataloged)

   If the dataset contains JCL, a DDNAME selection list will be displayed



        Press ENTER To Process Or END To Terminate The COPY Function
```

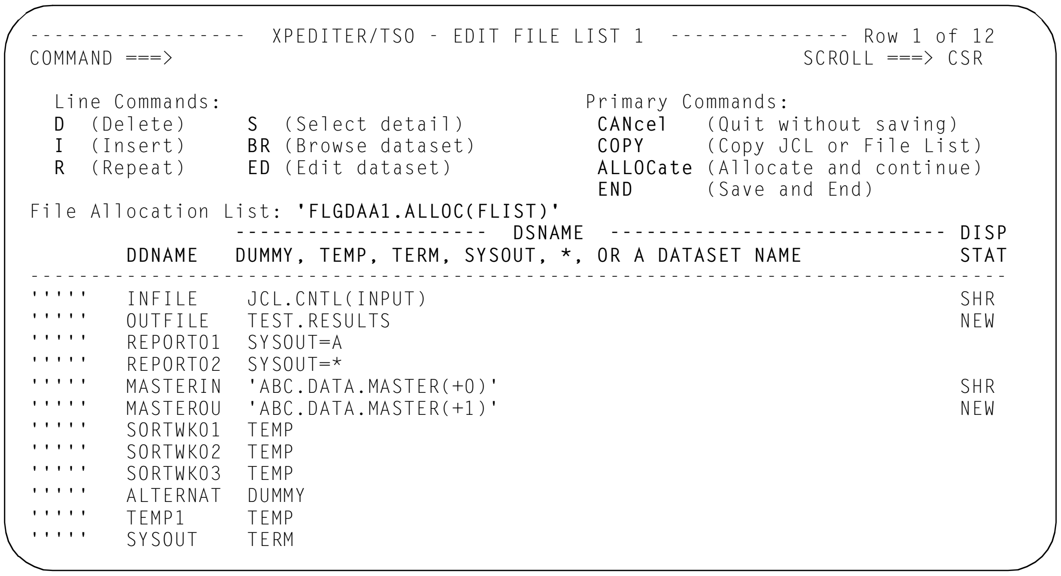
Figure A-6. Copy Screen—Specifying the DSNAME

The dataset is copied when you press **Enter** and the following occurs:

- If the dataset being copied contains a file list, the parameters associated with each file are listed on the appropriate Edit File List screens. The commands listed at the top of the screens can be used to browse and edit the data, copy data, allocate the file list, and so on. Refer to "Using the Edit File List Screens" on page A-5 for a description of how to use these screens.
- If the dataset being copied contains JCL, the JCL is displayed on the Select DDNAME screen. You can select the DD statements you want copied to the file list and perform other functions, such as editing. Refer to "Converting JCL to a File List" on page A-12.

Edit File List 1 Screen - DDNAME, DSNAME, and Disposition

The Edit File List 1 screen, shown in Figure A-7, is used to enter the ddname, dataset name, and disposition of the files to be allocated. When file information is copied from existing file lists, CLISTs, or JCL, this information is prefilled on this screen.



```
------------------  XPEDITER/TSO - EDIT FILE LIST 1  --------------- Row 1 of 12
COMMAND ===>                                                    SCROLL ===> CSR

  Line Commands:                              Primary Commands:
  D  (Delete)      S  (Select detail)          CANcel   (Quit without saving)
  I  (Insert)      BR (Browse dataset)         COPY     (Copy JCL or File List)
  R  (Repeat)      ED (Edit dataset)           ALLOCate (Allocate and continue)
                                               END      (Save and End)
File Allocation List: 'FLGDAA1.ALLOC(FLIST)'
                 ---------------------  DSNAME  --------------------------- DISP
        DDNAME   DUMMY, TEMP, TERM, SYSOUT, *, OR A DATASET NAME             STAT
--------------------------------------------------------------------------------
'''''   INFILE    JCL.CNTL(INPUT)                                            SHR
'''''   OUTFILE   TEST.RESULTS                                               NEW
'''''   REPORT01  SYSOUT=A
'''''   REPORT02  SYSOUT=*
'''''   MASTERIN  'ABC.DATA.MASTER(+0)'                                      SHR
'''''   MASTEROU  'ABC.DATA.MASTER(+1)'                                      NEW
'''''   SORTWK01  TEMP
'''''   SORTWK02  TEMP
'''''   SORTWK03  TEMP
'''''   ALTERNAT  DUMMY
'''''   TEMP1     TEMP
'''''   SYSOUT    TERM
```

Figure A-7. Edit File List 1 Screen—DDNAME, Dataset Name, and Disposition

- The following ddnames are **not** allowed:

```
JOBCAT       STEPLIB
JOBLIB       SYSPROC
STEPCAT
```

- Dataset names can be fully qualified and enclosed in quotes. If the name is not enclosed in quotes, the user's prefix is added at allocation time. A member of a PDS can be allocated by including the member name in parentheses with the dataset name. A temporary dataset that you want to allocate to more than one file can be entered in the form &&*name* (1-8 characters) with or without quotes.

  The following names can also be entered: DUMMY, TEMP, TERM, and SYSOUT. They refer to a dummy dataset, temporary dataset with no name, dataset sent to the terminal, and SYSOUT dataset, respectively.

  You can refer to the name of a previously allocated dataset by specifying *.*ddname*. The referenced ddname could have been allocated previously in the current file list or before the current file list was used.

  If you want to have the actual data stored in the file list, specify a dataset name of asterisk (*****). Refer to "In-Stream Data" on page A-16.

  The common types of datasets are discussed in "Types of Files That Can Be Allocated" on page A-15.

- Valid values for the DISP field are:

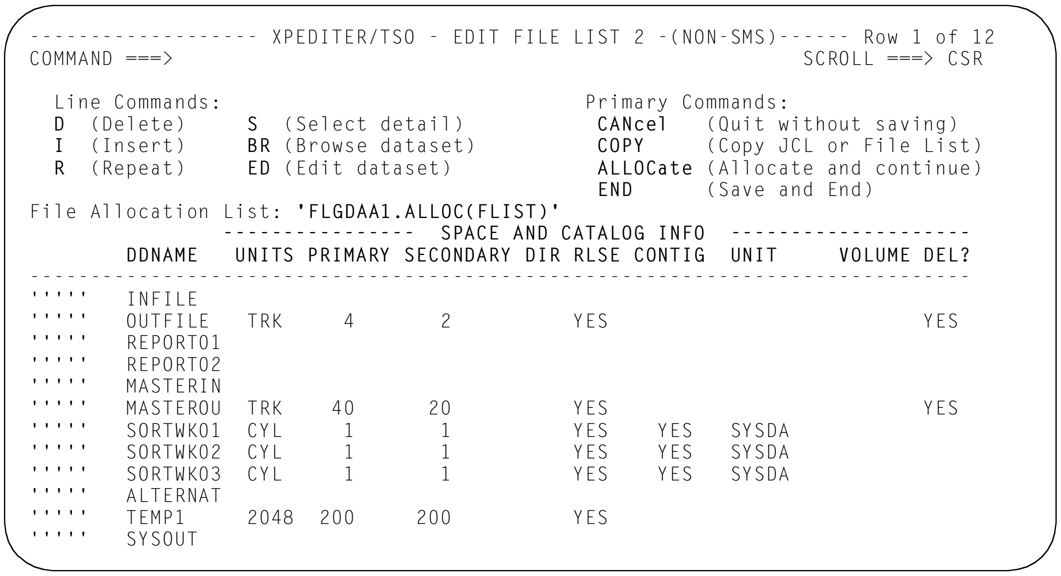
```
MOD (M)       OLD (O)
NEW (N)       SHR (S)
```

  For normal datasets, the default is SHR for temporary datasets, the default is NEW; for the other allocations (DUMMY, TERM, SYSOUT), the DISP field is forced to blank.

Edit File List 2 Screen - Space and Catalog Information

The Edit File List 2 screen shown in Figure A-8 is used for the space and catalog information required to allocate files with a disposition of NEW. If the file disposition is NEW, default values are prefilled for the UNITS, PRIMARY, SECONDARY, RLSE, and DEL? fields.

When file information is copied from existing file lists, CLISTs, or JCL, this information is prefilled on this screen.



```
------------------ XPEDITER/TSO - EDIT FILE LIST 2 -(NON-SMS)------ Row 1 of 12
COMMAND ===>                                                 SCROLL ===> CSR

  Line Commands:                               Primary Commands:
  D  (Delete)      S  (Select detail)           CANcel   (Quit without saving)
  I  (Insert)      BR (Browse dataset)          COPY     (Copy JCL or File List)
  R  (Repeat)      ED (Edit dataset)            ALLOCate (Allocate and continue)
                                                END      (Save and End)
File Allocation List: 'FLGDAA1.ALLOC(FLIST)'
                ---------------- SPACE AND CATALOG INFO  --------------------
        DDNAME   UNITS PRIMARY SECONDARY DIR RLSE CONTIG  UNIT     VOLUME DEL?
------------------------------------------------------------------------------
'''''   INFILE
'''''   OUTFILE   TRK     4       2          YES                          YES
'''''   REPORT01
'''''   REPORT02
'''''   MASTERIN
'''''   MASTEROU  TRK    40      20          YES                          YES
'''''   SORTWK01  CYL     1       1          YES    YES   SYSDA
'''''   SORTWK02  CYL     1       1          YES    YES   SYSDA
'''''   SORTWK03  CYL     1       1          YES    YES   SYSDA
'''''   ALTERNAT
'''''   TEMP1     2048  200     200          YES
'''''   SYSOUT
```

Figure A-8. Edit File List 2 Screen—Space and Catalog Information

Values for space and catalog information are:

UNITS
Valid values are TRACK (TRK, TRKS, TRA), CYL, or a block size of 0 to 65535 bytes.

PRIMARY
Number of units to allocate. Values are 0 to 99999.

SECONDARY
Space is allocated if the primary allocation is exceeded. Values are 0 to 99999.

DIR
If a new dataset is a PDS, this field reserves space for the member names. Values are 0 to 999.

RLSE
If the value is YES, all unused space is released when the dataset is closed. YES is the default for new or temporary datasets. Can be blank for NO.

CONTIG
If the value is YES, all primary space is allocated on contiguous tracks and cylinders when more than one cylinder is needed. Can be blank (default/NO).

UNIT
I/O unit—hardware address, device type, or group name. Enter if your installation requires it.

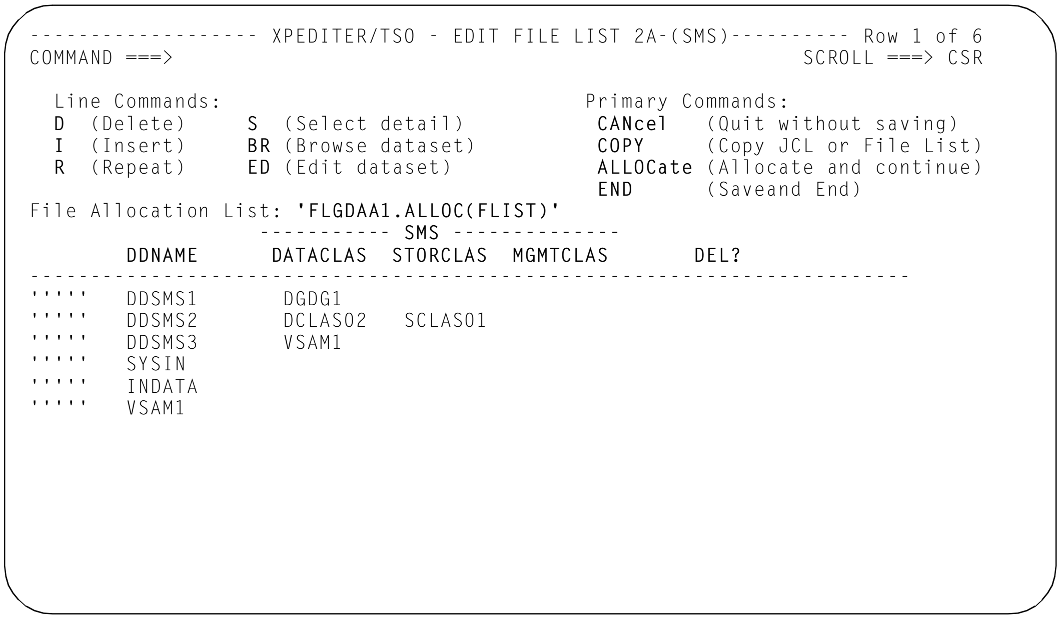
VOLUME
Required for an existing dataset that is not cataloged or if your installation requires it.

DEL?
Valid only if DISP=NEW. Values are YES (default), NO, USE, and question mark (?). If YES is specified, the old dsname is deleted before the new one is allocated. USE changes a DISP of NEW to OLD if the dataset exists. A question mark (?) causes you to be prompted for the desired action.

Edit File List 2A Screen - SMS Parameters

The Edit File List 2A screen shown in Figure A-9 is used to enter the SMS parameters (data class, storage class, and management class) associated with the ddname.

When file information is copied from existing file lists, CLISTs, or JCL, this information is prefilled on this screen.



```
------------------ XPEDITER/TSO - EDIT FILE LIST 2A-(SMS)---------- Row 1 of 6
COMMAND ===>                                                   SCROLL ===> CSR

  Line Commands:                                 Primary Commands:
  D  (Delete)      S  (Select detail)             CANcel   (Quit without saving)
  I  (Insert)      BR (Browse dataset)            COPY     (Copy JCL or File List)
  R  (Repeat)      ED (Edit dataset)              ALLOCate (Allocate and continue)
                                                  END      (Saveand End)
File Allocation List: 'FLGDAA1.ALLOC(FLIST)'
                         ----------- SMS -------------
        DDNAME       DATACLAS  STORCLAS  MGMTCLAS       DEL?
-------------------------------------------------------------------------
'''''   DDSMS1        DGDG1
'''''   DDSMS2        DCLAS02   SCLAS01
'''''   DDSMS3        VSAM1
'''''   SYSIN
'''''   INDATA
'''''   VSAM1
```

Figure A-9. Edit File List 2A Screen—SMS Parameters

Values for the SMS parameters are:

DATACLAS
Specifies the data class name defined by your site that contains the attributes related to the allocation of the file list.

STORCLAS
Specifies the storage class name defined by your site that contains the attributes related to the storage occupied by the file list.

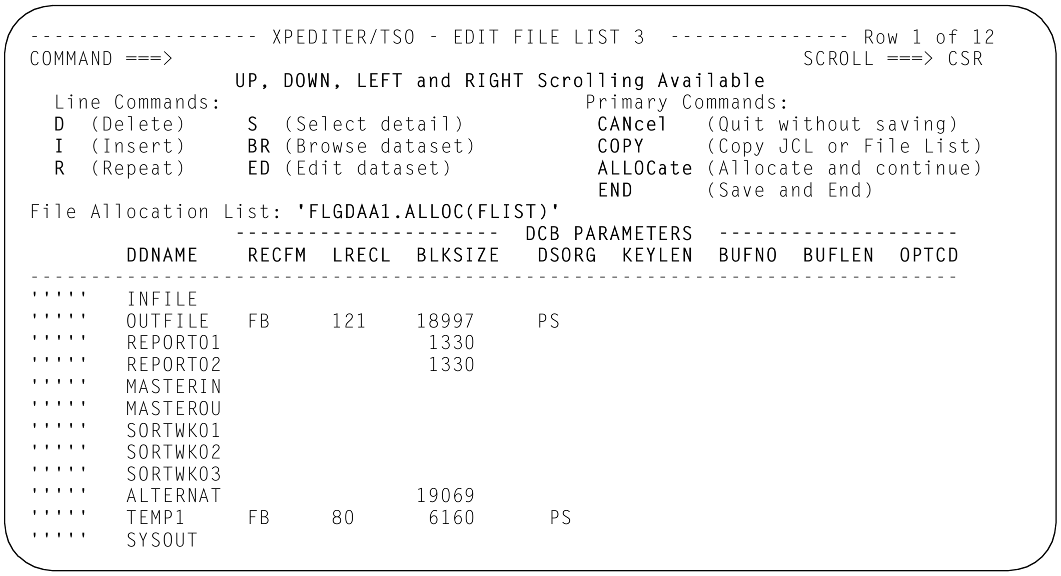
MGMTCLAS
Specifies the management class name defined by your site that specifies how SMS is to manage the dataset.

DEL?
Valid only if DISP=NEW. Values are YES (default), NO, USE, and question mark (?). If YES is specified, the old dsname is deleted before the new one is allocated. USE changes a DISP of NEW to OLD if the dataset exists. A question mark (?) causes you to be prompted for the desired action.

Edit File List 3 Screen - DCB Parameters

The Edit File List 3 screen in Figure A-10 is used to specify the DCB (data control block) parameters for each file listed.

When file information is copied from existing file lists, CLISTs, or JCL, this information is prefilled on this screen.



```
------------------ XPEDITER/TSO - EDIT FILE LIST 3  -------------- Row 1 of 12
COMMAND ===>                                                    SCROLL ===> CSR
                 UP, DOWN, LEFT and RIGHT Scrolling Available
  Line Commands:                                 Primary Commands:
  D  (Delete)      S  (Select detail)             CANcel   (Quit without saving)
  I  (Insert)      BR (Browse dataset)            COPY     (Copy JCL or File List)
  R  (Repeat)      ED (Edit dataset)              ALLOCate (Allocate and continue)
                                                  END      (Save and End)
File Allocation List: 'FLGDAA1.ALLOC(FLIST)'
                 ---------------------  DCB PARAMETERS  -------------------
        DDNAME    RECFM  LRECL  BLKSIZE   DSORG  KEYLEN  BUFNO  BUFLEN  OPTCD
-----------------------------------------------------------------------------
'''''   INFILE
'''''   OUTFILE   FB     121    18997     PS
'''''   REPORT01                 1330
'''''   REPORT02                 1330
'''''   MASTERIN
'''''   MASTEROU
'''''   SORTWK01
'''''   SORTWK02
'''''   SORTWK03
'''''   ALTERNAT                19069
'''''   TEMP1     FB     80      6160      PS
'''''   SYSOUT
```

Figure A-10. Edit File List 3 Screen—DCB Parameters

Values for the DCB parameters are:

RECFM
Record format. Valid values are:

F	FBM	VB	VBS
FB	FBS	VBA	VS
FBA	V	VBM	U

LRECL
Logical record length. Values are 0 to 32760. Do not specify a LRECL if the RECFM is **U** (undefined-length records).

Note: If the LRECL and BLKSIZE fields contain values, it is a good practice to specify RECFM; however, DSORG should be the default.

BLKSIZE
Block size. Values are 0 to 32760. If your program contains the clause `BLOCK CONTAINS 0 RECORDS`, enter a value to prevent an abend when the allocated file is opened. If fixed block, the value must be a multiple of the LRECL. If variable block, the value must be equal to or greater than the LRECL plus 4.

If the RECFM and LRECL fields are specified for a new dataset, a default value is provided for BLKSIZE.

DSORG
Dataset organization. Values are PS, PO, and DA. If DISP=NEW or TEMP, and DIR=0 or blank, PS is the default.

KEYLEN
Indicates the length of the record key. Values are 1 to 255.

BUFNO
Specifies the number of buffers to be used with the dataset. Values are 1 to 255.

BUFLEN
Indicates the buffer length. Values are 1 to 32760.

OPTCD
Optional system services. Valid values are:

A	F	Q	W
B	H	R	Z
C	J	T	
E	O	U	

Displaying File Parameters

Detailed information about the allocation, DCB, protection, and SYSOUT related parameters for a ddname in the file list can be displayed from any Edit File List screen.

To display detail information for a ddname, enter the S (Select detail) line command next to the ddname on the Edit File List screen. The File PARMS Menu shown in Figure A-11 is displayed.

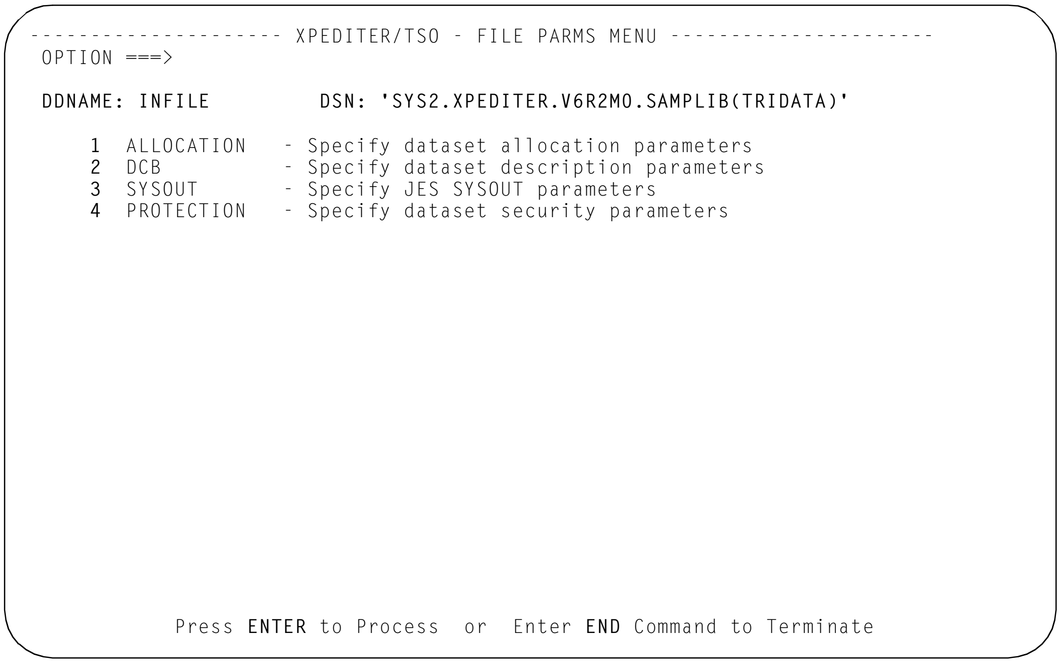

```
-------------------- XPEDITER/TSO - FILE PARMS MENU ---------------------
 OPTION ===>

 DDNAME: INFILE         DSN: 'SYS2.XPEDITER.V6R2M0.SAMPLIB(TRIDATA)'

     1  ALLOCATION   - Specify dataset allocation parameters
     2  DCB          - Specify dataset description parameters
     3  SYSOUT       - Specify JES SYSOUT parameters
     4  PROTECTION   - Specify dataset security parameters




            Press ENTER to Process  or  Enter END Command to Terminate
```

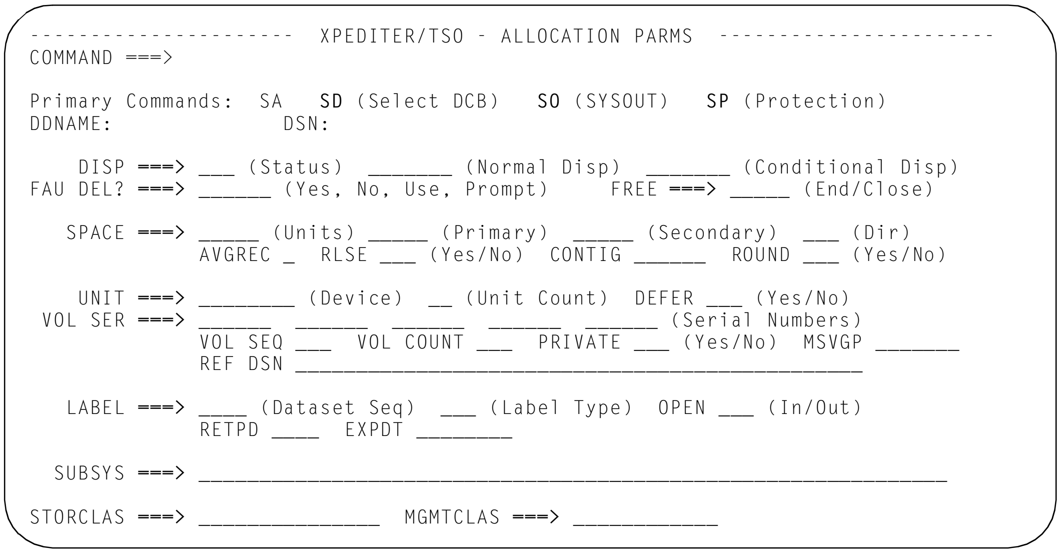
Figure A-11. File PARMS Menu

You can select to look at the allocation, DCB, protection, and SYSOUT parameters for the selected ddname. Depending on your selection, one of the four Parameters (PARMS) screens shown in Figure A-12 through Figure A-15 beginning on page A-11 is displayed. These screens display virtually all keywords/parameters supported by the JCL DD statements.

The primary commands (which are also used as panel codes) listed at the top of each screen can be used to display other PARMS screens.

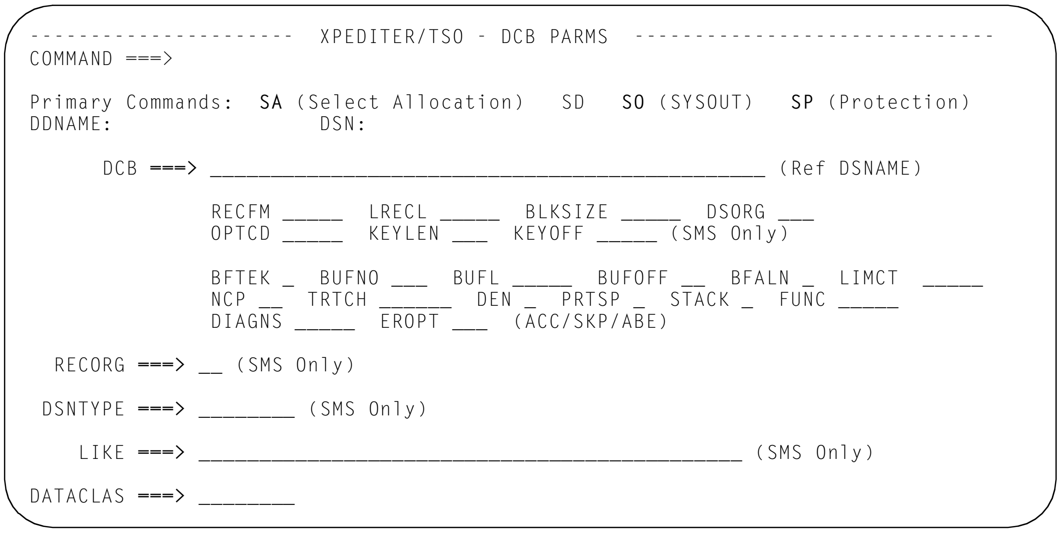
Shortcut: Use left and right scrolling to display the four screens in succession.

Items listed on the screens can be changed by typing over the information.



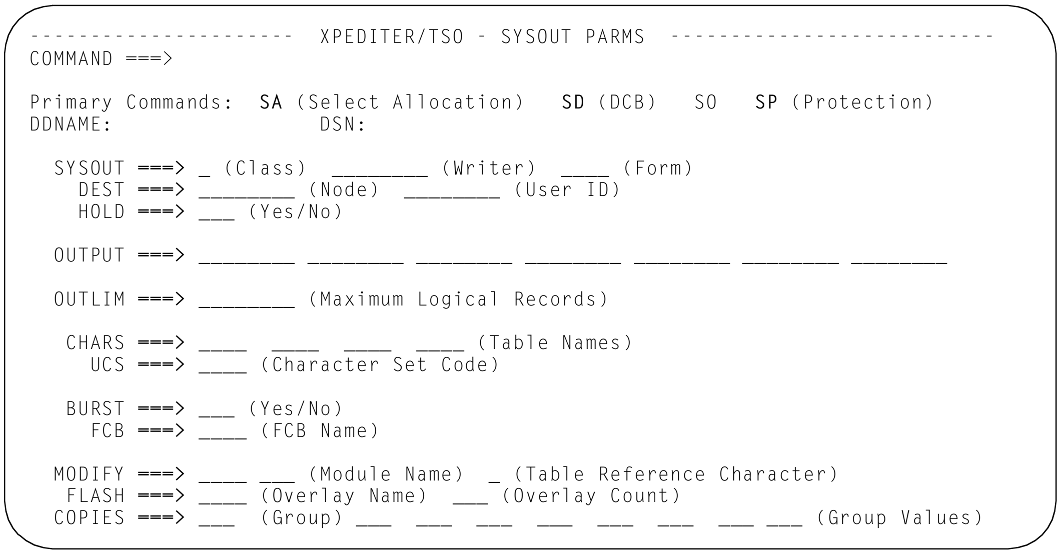
```
----------------------  XPEDITER/TSO - ALLOCATION PARMS  -----------------------
COMMAND ===>

Primary Commands:  SA   SD (Select DCB)   SO (SYSOUT)   SP (Protection)
DDNAME:               DSN:

    DISP ===> ___ (Status)  _______ (Normal Disp)  _______ (Conditional Disp)
FAU DEL? ===> ______ (Yes, No, Use, Prompt)     FREE ===> _____ (End/Close)

   SPACE ===> _____ (Units) _____ (Primary)  _____ (Secondary)  ___ (Dir)
              AVGREC _  RLSE ___ (Yes/No)  CONTIG ______  ROUND ___ (Yes/No)

    UNIT ===> ________ (Device)  __ (Unit Count)  DEFER ___ (Yes/No)
 VOL SER ===> ______  ______  ______  ______  ______ (Serial Numbers)
              VOL SEQ ___  VOL COUNT ___  PRIVATE ___ (Yes/No)  MSVGP _______
              REF DSN _______________________________________________

   LABEL ===> ____ (Dataset Seq)  ___ (Label Type)  OPEN ___ (In/Out)
              RETPD ____  EXPDT ________

  SUBSYS ===> ______________________________________________________________

STORCLAS ===> _______________  MGMTCLAS ===> ____________
```

Figure A-12. Allocation PARMS Screen (SA)



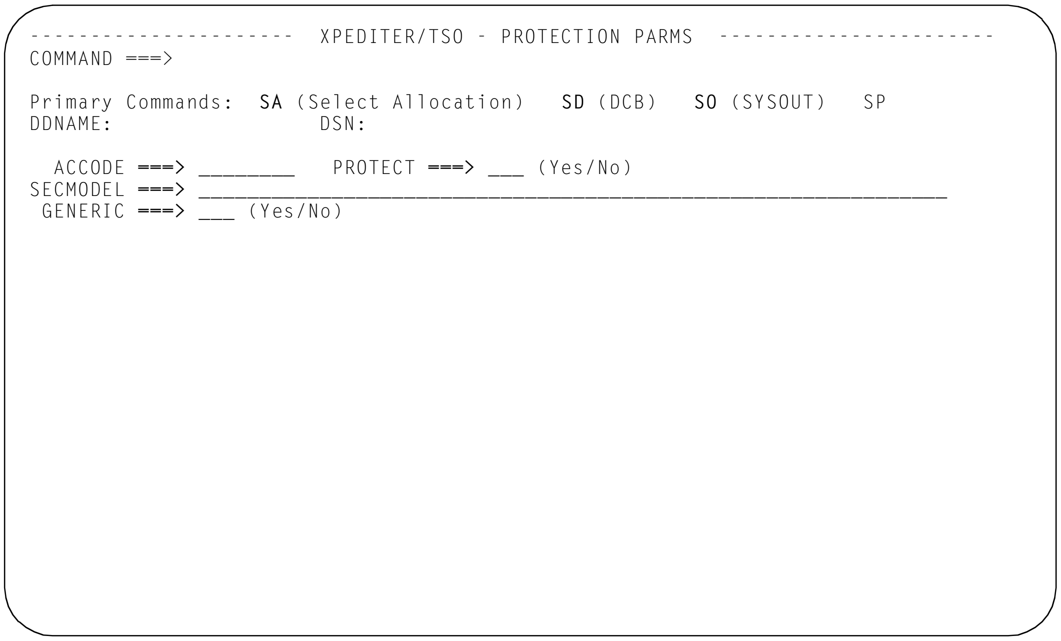
```
----------------------  XPEDITER/TSO - DCB PARMS  ------------------------------
COMMAND ===>

Primary Commands:  SA (Select Allocation)   SD   SO (SYSOUT)   SP (Protection)
DDNAME:                  DSN:

      DCB ===> ______________________________________________ (Ref DSNAME)

               RECFM _____  LRECL _____  BLKSIZE _____  DSORG ___
               OPTCD _____  KEYLEN ___  KEYOFF _____ (SMS Only)

               BFTEK _  BUFNO ___  BUFL _____  BUFOFF __  BFALN _  LIMCT  _____
               NCP __  TRTCH ______  DEN _  PRTSP _  STACK _  FUNC _____
               DIAGNS _____  EROPT ___  (ACC/SKP/ABE)

  RECORG ===> __ (SMS Only)

 DSNTYPE ===> ________ (SMS Only)

    LIKE ===> _____________________________________________ (SMS Only)

DATACLAS ===> ________
```

Figure A-13. DCB PARMS Screen (SD)



```
---------------------  XPEDITER/TSO - SYSOUT PARMS  --------------------------
COMMAND ===>

Primary Commands:  SA (Select Allocation)   SD (DCB)   SO   SP (Protection)
DDNAME:                 DSN:

  SYSOUT ===> _ (Class)  ________ (Writer)  ____ (Form)
    DEST ===> ________ (Node)  ________ (User ID)
    HOLD ===> ___ (Yes/No)

  OUTPUT ===> ________ ________ ________ ________ ________ ________ ________

  OUTLIM ===> ________ (Maximum Logical Records)

   CHARS ===> ____  ____  ____  ____ (Table Names)
     UCS ===> ____ (Character Set Code)

   BURST ===> ___ (Yes/No)
     FCB ===> ____ (FCB Name)

  MODIFY ===> ____ ___ (Module Name)  _ (Table Reference Character)
   FLASH ===> ____ (Overlay Name)  ___ (Overlay Count)
  COPIES ===> ___  (Group) ___  ___  ___  ___  ___  ___  ___ ___ (Group Values)
```

Figure A-14. SYSOUT PARMS Screen (SO)



```
---------------------  XPEDITER/TSO - PROTECTION PARMS  ----------------------
COMMAND ===>

Primary Commands:  SA (Select Allocation)   SD (DCB)   SO (SYSOUT)   SP
DDNAME:                 DSN:

  ACCODE ===> ________   PROTECT ===> ___ (Yes/No)
SECMODEL ===> ______________________________________________________________
 GENERIC ===> ___ (Yes/No)
```

Figure A-15. Protection PARMS Screen (SP)

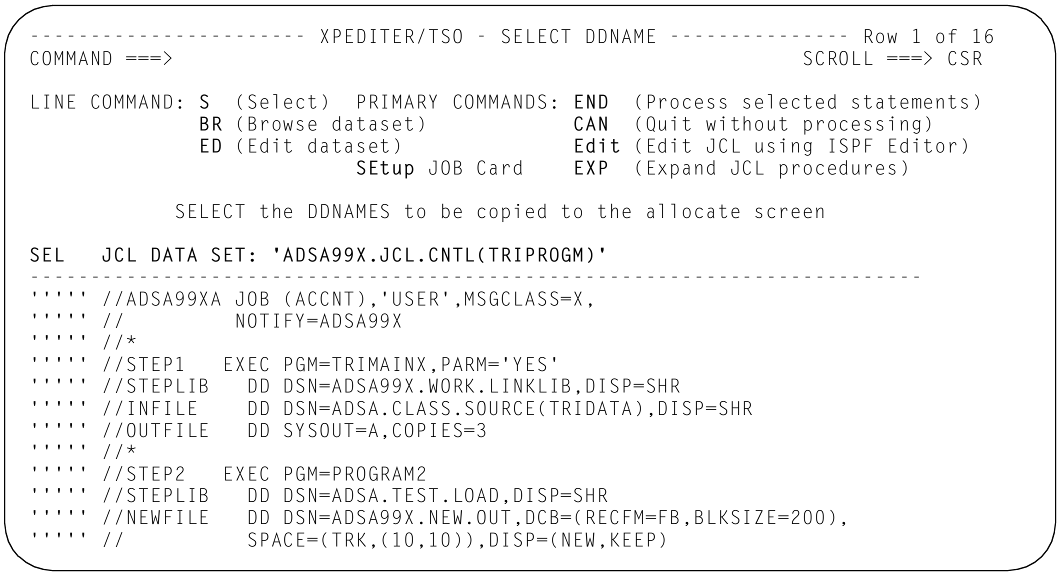
Converting JCL to a File List

When you specify a dataset that contains JCL, the dataset is displayed on the Select DDNAME screen shown in Figure A-16. From this data, you can select the ddnames to be converted to a file list. The parameters (space and catalog or SMS and DCB) associated with each selected ddname are also copied to the appropriate Edit File List screens.

The primary commands UP and DOWN or PF keys can be used to scroll the screen.

Note: The primary commands listed on the screen may vary.

Information on the screen can be typed over; e.g., commented lines can be changed to uncommented lines and the I(nsert), R(epeat), and D(elete) line commands are also available when editing.



```
---------------------- XPEDITER/TSO - SELECT DDNAME --------------- Row 1 of 16
COMMAND ===>                                                   SCROLL ===> CSR

LINE COMMAND: S  (Select)  PRIMARY COMMANDS: END  (Process selected statements)
              BR (Browse dataset)            CAN  (Quit without processing)
              ED (Edit dataset)              Edit (Edit JCL using ISPF Editor)
                          SEtup JOB Card     EXP  (Expand JCL procedures)

            SELECT the DDNAMES to be copied to the allocate screen

SEL   JCL DATA SET: 'ADSA99X.JCL.CNTL(TRIPROGM)'
--------------------------------------------------------------------------
''''' //ADSA99XA JOB (ACCNT),'USER',MSGCLASS=X,
''''' //          NOTIFY=ADSA99X
''''' //*
''''' //STEP1   EXEC PGM=TRIMAINX,PARM='YES'
''''' //STEPLIB   DD DSN=ADSA99X.WORK.LINKLIB,DISP=SHR
''''' //INFILE    DD DSN=ADSA.CLASS.SOURCE(TRIDATA),DISP=SHR
''''' //OUTFILE   DD SYSOUT=A,COPIES=3
''''' //*
''''' //STEP2   EXEC PGM=PROGRAM2
''''' //STEPLIB   DD DSN=ADSA.TEST.LOAD,DISP=SHR
''''' //NEWFILE   DD DSN=ADSA99X.NEW.OUT,DCB=(RECFM=FB,BLKSIZE=200),
''''' //           SPACE=(TRK,(10,10)),DISP=(NEW,KEEP)
```

Figure A-16. Select DDNAME Screen

Use the primary commands as follows:

END	Copies the selected ddnames. After you enter the END command, but before the files are copied, any errors are reported. The errors must be corrected before the copy is performed.
CANcel	Cancels the ddname selection.
Edit	Accesses an ISPF edit session where you can make changes to your JCL.
EXPand	Expands references to cataloged, in-stream procedures and performs symbolic parameter replacement, making procedures available for use. See "Things to Know About JCL Expansion" on page A-14 for more information.
SEtup	Displays a screen on which you can enter and change the skeleton job card needed for the EXPAND function.

Use the line commands as follows:

S (Select)	Selects the DD statements to be copied to the file list. When you select DD statements, you are basically selecting the file names and all concatenated DD statements to be copied as a unit. Enter S on a line to select a single statement. Enter a beginning and ending SS to select all file names between the two entries. When you enter an S for an EXEC statement, all file names up to the next EXEC statement are selected. Any combination of these entries can be used to select all the files you require.
BR (Browse)	Displays the dataset associated with the selected statement.
ED (Edit)	Displays the dataset associated with the selected statement for editing.

Things to Know About JCL Expansion

There are two ways to expand the JCL:

1. In one mode (job submission), the EXPAND command submits the current JCL as a job. If there are multiple jobs in the JCL, only the first job is processed; therefore, you should delete any jobs preceding the one you want to process.

   If there is a job card present in the JCL, it is not used for this process. The SETUP command can be used to look at and change the job card that is used. Consider the following:

   - MSGCLASS, MSGLEVEL, and TYPRUN are supplied by the expand process and should **not** be included on the job card.

- The job card should be valid to avoid job-not-found or no-held-output conditions.
- Include only the minimum parameters required to pass the installation validation routines. Since the job does not actually run, extra parameters tend to cause undesired results, such as extra asynchronous messages to the terminal.

2. In the other mode (direct expansion), XPEDITER directly accesses the procedure libraries. For this to be possible, the installer must have provided the list of datasets containing the catalogued procedures and you must have read access to all of these libraries.

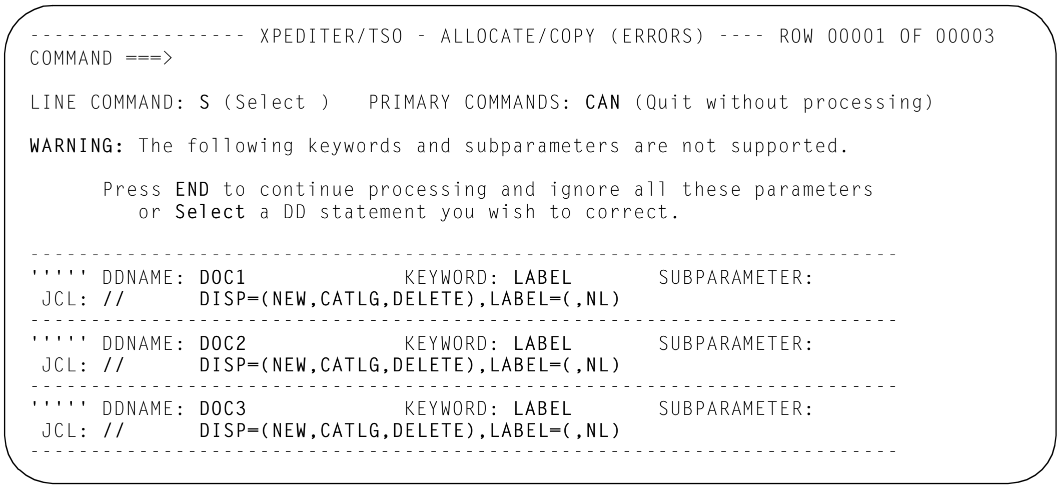
Unsupported Keywords and Subparameters

Not all JCL parameters in the JCL dataset are copied to the file list; DEN is one, for example. The following JCL items are not supported:

- Special ddnames - JOBCAT, JOBLIB, STEPCAT, STEPLIB, SYSPROC
- Datasets extending onto more than five volumes
- Unresolved symbolic references
- Parameters not corresponding to fields on the Edit File List screens.

If a selected ddname contains parameters/keywords not specified on the Edit File List screens, the warning screen in Figure A-17 is displayed.

Errors are identified by ddname, keyword, and JCL statement. If an error is found on a concatenated file, the name appears in the form *ddname+nn*.



```
----------------- XPEDITER/TSO - ALLOCATE/COPY (ERRORS) ---- ROW 00001 OF 00003
COMMAND ===>

LINE COMMAND: S (Select )   PRIMARY COMMANDS: CAN (Quit without processing)

WARNING: The following keywords and subparameters are not supported.

      Press END to continue processing and ignore all these parameters
         or Select a DD statement you wish to correct.

-----------------------------------------------------------------------
''''' DDNAME: DOC1             KEYWORD: LABEL       SUBPARAMETER:
 JCL: //      DISP=(NEW,CATLG,DELETE),LABEL=(,NL)
-----------------------------------------------------------------------
''''' DDNAME: DOC2             KEYWORD: LABEL       SUBPARAMETER:
 JCL: //      DISP=(NEW,CATLG,DELETE),LABEL=(,NL)
-----------------------------------------------------------------------
''''' DDNAME: DOC3             KEYWORD: LABEL       SUBPARAMETER:
 JCL: //      DISP=(NEW,CATLG,DELETE),LABEL=(,NL)
-----------------------------------------------------------------------
```

Figure A-17. Allocate/Copy (Errors) Screen

The errors can be ignored by entering **END**. The JCL is displayed on the Edit File List 1 screen. To correct the errors, enter the S line command next to the ddname in question. The Select DDNAME 1 screen is displayed with the selected ddname at the top of the screen and the cursor positioned at the value in error. You can correct the errors from this screen.

By entering **CANCEL**, you are returned to the screen from which you entered the FAU.

Types of Files That Can Be Allocated

This section describes some common types of datasets that can be allocated using the File Allocation Utility.

Existing Datasets

Existing datasets can be allocated with a disposition of SHR, OLD, or MOD. Unless exclusive use of a dataset is required for your application, SHR is the recommended disposition.

A dataset allocation with the disposition of SHR, OLD, or MOD fails if:

1. You request disposition OLD/MOD, but the dataset is already allocated to another user.
2. You request disposition SHR, but the dataset is already allocated exclusively OLD/MOD to another user.
3. The dataset is not found (SHR or OLD only).

Notes:

1. For uncataloged datasets, enter the name of the volume on which the dataset resides in the VOLUME field on the Edit File List 2 screen.
2. For concatenating datasets, 1) list the dataset with the largest block size first; 2) code the dsnames of the datasets to be concatenated on successive lines, leaving the DDNAME field blank after the first line; and 3) concatenate a maximum of 255 sequential or 16 partitioned datasets.

New Datasets

To allocate a new dataset, values must be supplied for the DDNAME, DSNAME, DISP (NEW), UNITS, PRIMARY, and SECONDARY fields on the Edit File List screens. A value must be entered in the DIR field if you are allocating a PDS. If no value is entered in the DIR field, the DSORG is assumed to be PS (physical sequential).

Values must also be supplied for the RECFM, LRECL, and BLKSIZE fields unless these parameters are coded in the program itself. Some versions of MVS will calculate an optimum blocksize. For these systems, the BLKSIZE field should be left blank or set to zero. If your version of MVS does not provide this, the FAU will calculate an optimum block size (based on an installation specification) if you enter the RECFM and LRECL and leave the BLKSIZE blank.

Warning:
If a COBOL program specifies the size of the physical record (BLKSIZE) in the BLOCK CONTAINS 0 RECORDS clause of the FD entry, the BLKSIZE field should be specified to avoid a run-time abend.

For a new or temporary dataset, values are prefilled for the UNITS, PRIMARY, SECONDARY, RLSE, and DEL? fields. YES is entered in the RLSE field to release unused space when the dataset is closed. Entering YES in the DEL? field tells XPEDITER/TSO to delete the old file, if any, when attempting to allocate a new dataset.

If your installation requires it, a value must be specified for the UNIT and VOLUME fields on the Edit File List 2 (space and catalog information) screen.

Dummy Files

Dummy files are generally used for nonessential input and output. These files can also be used to test program flow without actually processing data. For instance, unwanted output can be suppressed by giving the output datasets a dummy status.

Since dummy files do not exist, DCB requirements are the same as for a new dataset. However, since no real allocation occurs, space information is not used.

Temporary Files

Temporary datasets are often used to allocate sort or other work files. Since a temporary file usually has no dsname, the data it contains is lost whenever the file is freed or when you log off.

Temporary datasets defined with the TEMP keyword have a system-generated, unique name. They must generally be allocated as new datasets. Temporary datasets using an &&*name* form also have a system-generated name. They cannot be cataloged and are automatically deleted when the allocation creating them is freed. Thus, they can only be allocated as OLD while the allocation creating them still exists.

Space and DCB requirements are the same as for other new (or existing) datasets.

In-Stream Data

The data for the program can be provided as part of the file list. To use this feature, specify an asterisk (*) as the dataset name for the data. You can then use the EDIT and BROWSE commands to create, modify, and review the data associated with the allocations.

Since these datasets are temporary, the allocation and DCB parameters are the same as for temporary datasets. Additionally, since the data is manipulated via the ISPF/PDF editor, the DCB information must be consistent with ISPF/PDF (fixed or variable length records and LRECL of 0 to 255 bytes). Also, for compatibility with normal JCL conventions, the RECFM defaults to FB and the LRECL defaults to 80. If variable length records are specified, LRECL defaults to 84.

Allocating Files to the Terminal

Sequential files can be allocated to the terminal. SYSOUT files (COBOL DISPLAY verb) are often allocated to the terminal. The files can be opened for input or output. Unlike other forms of new datasets, a block size is not required.

If an input file is allocated to the terminal, XPEDITER/TSO suspends the execution of your program while awaiting terminal input. The keyboard is unlocked and XPEDITER prompts you to enter data. A slash asterisk (/*) is entered to indicate the end of the file.

SYSOUT Files

The SYSOUT parameter instructs the system to queue the output on a direct access volume, and system output writers later transcribe the output onto the specified I/O device. To allocate a SYSOUT file, enter a ddname in the DDNAME field and the SYSOUT=*class parameter* in the DSNAME field. The SYSOUT class can be any alphanumeric (A-Z, 0-9) character or an asterisk (*) - default. As with a terminal file, a block size is not required.

Generation Datasets

A generation data group (GDG) is a collection of cataloged datasets that have the same name and are related to one another chronologically. A GDG can consist of sequential, partitioned, and direct datasets residing on direct access volumes. The GDG dsname is limited to 35 characters. Each generation can be distinguished from the others by its relative or absolute generation number. A relative generation number cannot exceed 255.

To create or retrieve a generation dataset, enter the GDG name, followed by its relative (or absolute) number in the DSNAME field. If a relative generation number is used, you cannot also specify a member name.

If GDG stability was set to YES during the installation process, the relative generation numbers remain static during a TSO session. Specifically, if you create a new generation with a relative generation number of +1 and you want to refer to it later during your session, continue to refer to it as +1.

If GDG stability was set to NO during the installation process, the positive relative numbers will always allocate a new dataset relative to the current generation, notwithstanding any prior use of the same generation.

This stability has implications when multiple new generations are allocated in TSO, but are **not** allocated in ascending order. You must assume that a GDG has 25 generations at TSO LOG and the following allocations have been made:

```
ALLOCATE              STABILITY=NO                 STABILITY=YES
 GDG(+2)          creates G0027V00  (25+2)       allocates G0027V00
 GDG(+1)          creates G0028V00  (27+1)       allocates G0026V00
 GDG(0)         allocates G0028V00  (28+0)       allocates G0025V00
```

On the DCB PARMS screen, enter the name of a fully qualified dataset where XPEDITER/TSO can obtain the model DCB information, unless a model DSCB was created for the generation data group. If a PDS is specified, do not include a member name.

If all the generations in a GDG have identical DCB attributes, the generations can be retrieved together as a single dataset. To process the entire GDG, enter its dsname without a relative generation number.

ISAM Files

ISAM files can be allocated, but certain IBM SVC 99 restrictions for ISAM files also apply:

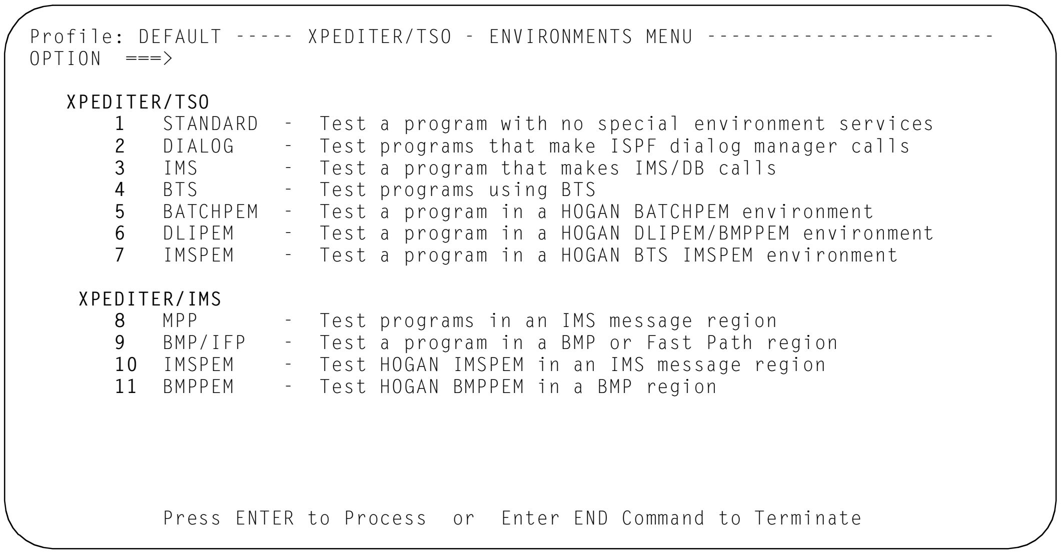
- The data and key areas must be in the same dataset, on one volume.
- The file cannot be NEW.

Appendix B. XPEDITER/TSO Environment Test Screens

The first time you invoke an XPEDITER test session using option 2 (TSO) on the Primary Menu, the XPEDITER/TSO Environments Menu shown in Figure B-1 is displayed.

Note: The range of environment options available on this menu depends on the site defaults set by your installer.

Each option on the Environments Menu accesses a test screen. The environment test screens are used to specify the name of the program to be tested and other parameters associated with the program and the test session. For example, if you are testing a program in a standard environment with no special services, you would select option 1 (Standard). The Standard test screen shown in Figure B-3 on page B-4 is displayed.



```
Profile: DEFAULT ----- XPEDITER/TSO - ENVIRONMENTS MENU ------------------------
OPTION  ===>

   XPEDITER/TSO
       1   STANDARD  -  Test a program with no special environment services
       2   DIALOG    -  Test programs that make ISPF dialog manager calls
       3   IMS       -  Test a program that makes IMS/DB calls
       4   BTS       -  Test programs using BTS
       5   BATCHPEM  -  Test a program in a HOGAN BATCHPEM environment
       6   DLIPEM    -  Test a program in a HOGAN DLIPEM/BMPPEM environment
       7   IMSPEM    -  Test a program in a HOGAN BTS IMSPEM environment

    XPEDITER/IMS
       8   MPP       -  Test programs in an IMS message region
       9   BMP/IFP   -  Test a program in a BMP or Fast Path region
       10  IMSPEM    -  Test HOGAN IMSPEM in an IMS message region
       11  BMPPEM    -  Test HOGAN BMPPEM in a BMP region




           Press ENTER to Process  or  Enter END Command to Terminate
```

Figure B-1. XPEDITER/TSO Environments Menu

The structure diagram in Figure B-2 on page B-2 shows the menu hierarchy of the execution environments you can select.

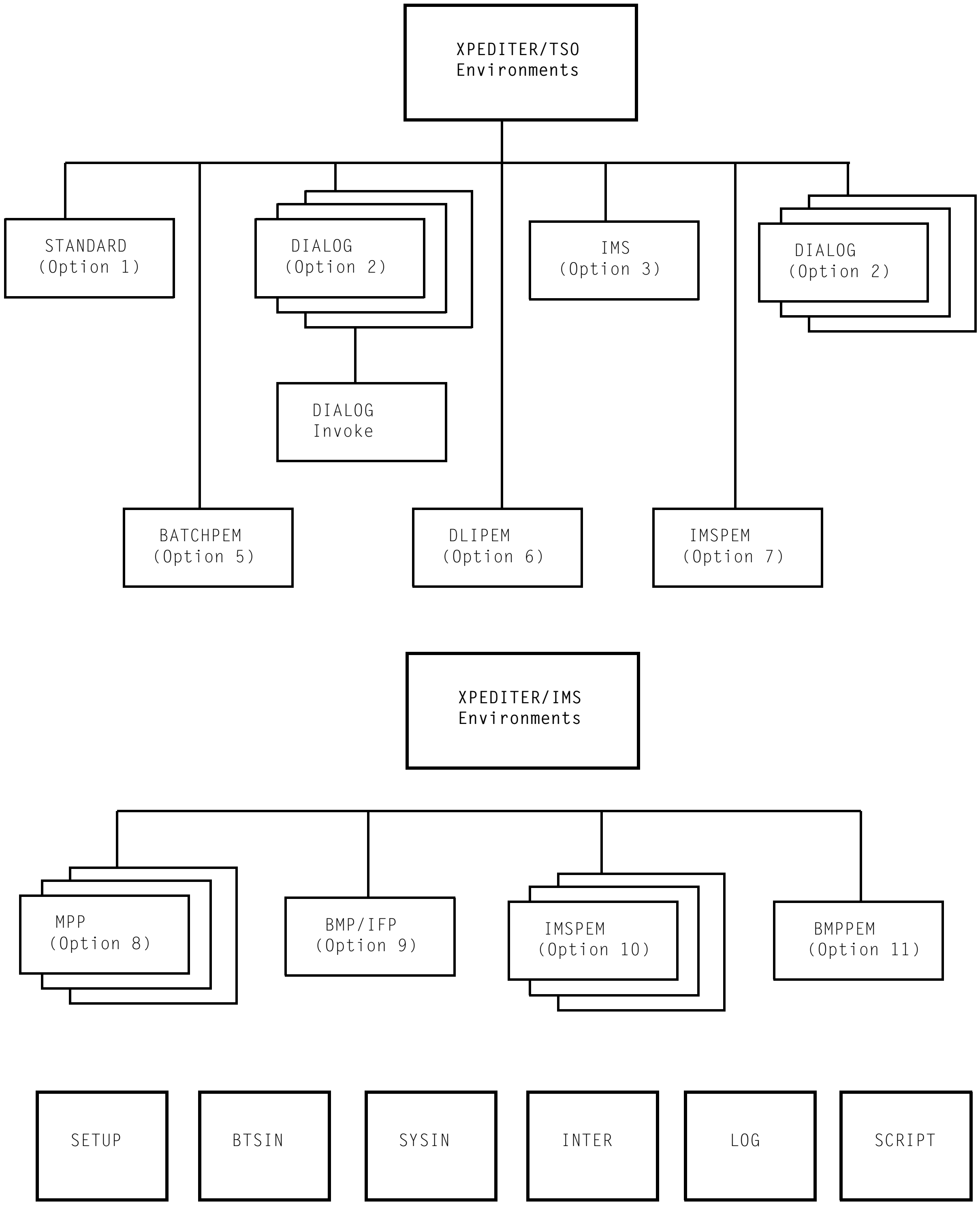


Figure B-2. Diagram of the Execution Environments

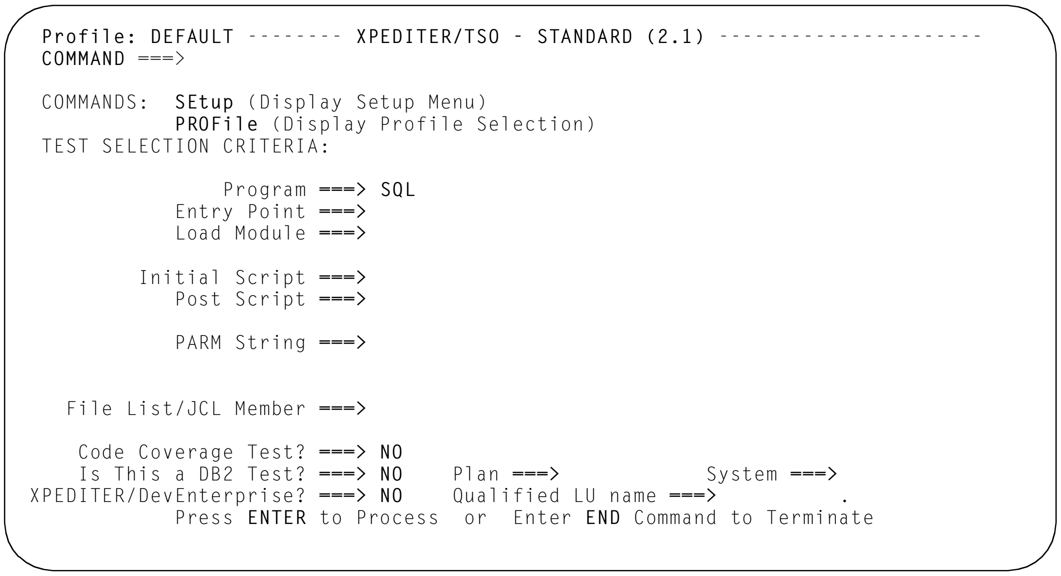
Each option in Figure B-2 represents the test screen for the environment. The test screens let you specify the environment parameters for initiating a debugging session.

Some test screens have commands that will access submenus. The commands that may appear on a test screen are described below:

SE (SETUP)	Displays the Setup Menu. The options on the Setup Menu display submenus that are used to override the default values defined at your site. You can specify the load libraries, DDIO files, test script libraries, session log and test script dataset disposition, DSN libraries, and PANEXEC load libraries. The SETUP command is valid on all test screens.
PROFile (PROFILE)	Displays a list of current user profiles that retain information about XPEDITER test environments. You may Select (S), Delete (D), Copy (C), Rename (R), or Use (U) the profiles stored in the displayed file by entering the appropriate line command in the CMD (Command) field. You may also utilize the only Primary Command, the Merge command. This command lets you copy other user's profiles to your own profile.
L, LO (LOG)	Browses the session log. The log file contains a record of the commands that were entered during the debugging session and the responses to them. The LOG command is valid on all the test screens. However, the log is not displayed until *after* you execute a program.
SC (SCRIPT)	Browses and edits the test script created during the debugging session. A sequential test script file contains commands recorded during the debugging session. The generated test script must be moved to a member of an INCLUDE library for it to be executed. The SCRIPT command is valid on all test screens, but the script is not displayed until *after* you execute a program.
BT (BTSIN)	Displays a list of BTS transactions that you can select to debug. Any transactions selected through the use of the BTSIN command are automatically entered on the BTS or Hogan IMSPEM screen. The BTSIN command is valid only on the BTS and Hogan IMSPEM test screens.
SY (SYSIN)	Displays the Hogan SYSIN dataset, so you can edit the control cards. The SYSIN command is valid only on the Hogan test screens.
IN (INTER)	Displays a list of transactions that are intercepted by XPEDITER/IMS.
UP (UP)	Scrolls toward the top of your screen. Scrolling up rolls the screen downward, bringing the previous section of the data into view.
DO (DOWN)	Scrolls toward the end of your screen. Scrolling down rolls the screen upward, bringing the following section of the data into view.

Standard Test Screen – Environments Menu - Option 1

The Standard Test screen (Environments Menu) shown in Figure B-3 on page B-4 is used to specify the standard environment parameters. Batch programs that process QSAM or VSAM files, batch programs that issue third-party database calls (IMS/DB, ADABAS, TOTAL, TIS, SUPRA, DATACOM/DB, and System 2000), and batch programs that issue EXEC SQL (DB2) statements can be executed in this environment.



```
 Profile: DEFAULT -------- XPEDITER/TSO - STANDARD (2.1) ----------------------
 COMMAND ===>

 COMMANDS:  SEtup (Display Setup Menu)
            PROFile (Display Profile Selection)
 TEST SELECTION CRITERIA:

                Program ===> SQL
            Entry Point ===>
            Load Module ===>

         Initial Script ===>
            Post Script ===>

            PARM String ===>


   File List/JCL Member ===>

    Code Coverage Test? ===> NO
    Is This a DB2 Test? ===> NO    Plan ===>            System ===>
XPEDITER/DevEnterprise? ===> NO    Qualified LU name ===>          .
            Press ENTER to Process  or  Enter END Command to Terminate
```

Figure B-3. Standard Environment Screen

The fields on the Standard Environment screen are described below:

Program
Required. Enter the name of the load module.

Note: If executing other than the linked entry name, then the program name is the CSECT name and the load module name must be specified under Load Module.

Entry Point
Enter an alternate entry point if execution is to start at a point other than the link-edited entry point.

Load Module
Enter only if executing a CSECT other than the link-edited entry point of a load module, such as unit testing CSECTs within a load module.

Initial Script
Enter the member name of a test script if you want to execute a predefined command stream at the *beginning* of the debugging session. This member will be executed after the inclusion of the Site-wide script member @@SITE@@, if defined.

Post Script
Enter the member name of a test script if you want to execute a predefined command stream at the *end* of the debugging session.

PARM String
Enter the parameter string if the program (and any called programs) expects a run-time parameter. You can enter up to 100 characters in this field.

If the run-time parameter consists of several substrings separated by commas, or if it contains special characters (/, =, etc.), enclose the entire parameter in quotes (single or double).

File List/JCL Member
Enter the dataset name that contains the file list, CLIST, or JCL. The File Allocation Utility (FAU) preallocates files and databases that are processed by the program upon entry to the debugging session.

If the member name of a PDS is omitted, a member list is displayed.

If the dataset contains a CLIST, XPEDITER/TSO immediately executes it and begins the debugging session.

If the dataset contains a file list or JCL, the FAU is invoked to dynamically allocate the specified files before beginning the debugging session. Refer to Appendix A, “Using the File Allocation Utility” on page A-1 for detailed information about the FAU.

Code Coverage Test?
Enter **YES** if XPEDITER/Code Coverage data should be collected for this test. The default value is NO.

Note: The Code Coverage option is only available if you have purchased XPEDITER/Code Coverage.

Is This a DB2 Test?
Enter **YES** if the program executes EXEC SQL statements. XPEDITER/TSO executes the DSN RUN command and establishes the DB2 environment upon entry to the debugging session. The default value is NO.

Plan
Enter the DB2 plan name generated during the bind process. If omitted, the plan name defaults to the name specified in the Program field.

System
Enter the DB2 subsystem name. The subsystem name depends on the release level of DB2 allocated to the DSNLOAD library specified on the Setup option.

XPEDITER/DevEnterprise?
Enter **YES** in this field if you want to connect to XPEDITER/DevEnterprise. The default is NO.

Qualified LU name
Enter the fully qualified LU name if you are going to use XPEDITER/DevEnterprise. Enter the same name that was entered on the installation screen.

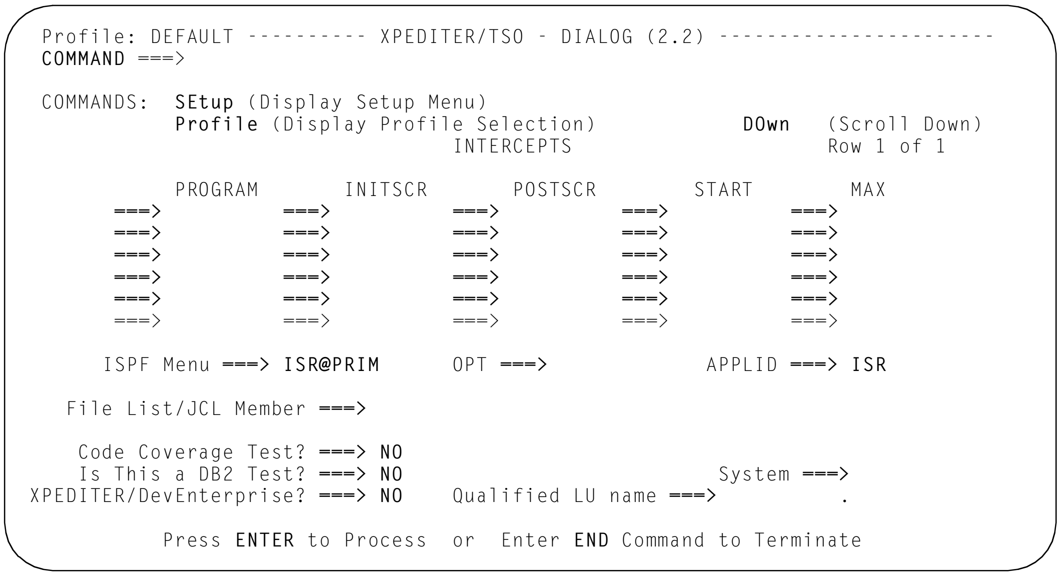
Press **Enter** to initiate the debugging session. Refer to Chapter 5, "Debugging Interactively" on page 5-1 for information on how to debug your program.

Dialog Test Screen – Environments Menu - Option 2

The Dialog Test screen, shown in Figure B-4 on page B-6, lets you specify the dialog environment parameters and initiate a debugging session. The screen is designed to allow programmers to debug ISPF dialog programs that are running as a part of the larger application process. For instance, programs that are invoked by entering data from an ISPF panel, issuing an edit macro, or executing a CLIST can be debugged in this environment.

All the dialog components, such as screens, CLISTs, tables, messages, and load modules, must be preallocated before initiating the debugging session unless the dialog includes a function to use the ISPF LIBDEF facility. Refer to the IBM *ISPF Dialog Management Services Manual* for additional information regarding file allocations for dialog testing.

Note: The load module libraries must be allocated to ISPLLIB and be included in the XPEDITER load library list.



```
 Profile: DEFAULT ---------- XPEDITER/TSO - DIALOG (2.2) -----------------------
 COMMAND ===>

 COMMANDS:  SEtup (Display Setup Menu)
            Profile (Display Profile Selection)           DOwn   (Scroll Down)
                                 INTERCEPTS                      Row 1 of 1

            PROGRAM       INITSCR       POSTSCR        START        MAX
       ===>          ===>          ===>          ===>          ===>
       ===>          ===>          ===>          ===>          ===>
       ===>          ===>          ===>          ===>          ===>
       ===>          ===>          ===>          ===>          ===>
       ===>          ===>          ===>          ===>          ===>
       ===>          ===>          ===>          ===>          ===>

      ISPF Menu ===> ISR@PRIM      OPT ===>             APPLID ===> ISR

   File List/JCL Member ===>

    Code Coverage Test? ===> NO
    Is This a DB2 Test? ===> NO                           System ===>
XPEDITER/DevEnterprise? ===> NO    Qualified LU name ===>          .

          Press ENTER to Process  or  Enter END Command to Terminate
```

Figure B-4. Dialog Test Screen

The fields on the Dialog Test screen are described below.

PROGRAM

Required. Enter a series of load module names that you want to debug when the dialog is invoked. You must specify either the main program or subprograms. Do not specify both.

Six load modules can be specified on this screen. If you want to continue the list, you can scroll down to a subsequent screen.

INITSCR

Enter the member name of a test script if you want to execute a predefined command stream at the *beginning* of the debugging session. This member will be executed after the inclusion of the Site-wide script member @@SITE@@, if defined.

POSTSCR

Enter the member name of a test script if you want to execute a predefined command stream at the *end* of the debugging session.

START

Enter up to four digits to specify on which occurrence of the program invocation the intercept is to begin. By default, program intercept begins on the first occurrence. When the program is intercepted, the debugging session is initiated and the source is displayed.

MAX

Enter up to four digits to specify the maximum number of times the program intercept is processed. If this field is left blank, the value defaults to an infinite number. If you enter an EXIT command and there are still intercepts remaining, those intercepts are ignored.

ISPF Menu

Required. Specify the driver panel to start your dialog application. Typically, this is the ISPF/PDF Primary Option Menu, ISR@PRIM. You can initiate your dialog application through the dialog test facilities (option 7.1 of ISPF/PDF) and trace the ISPF variables while tracing the host variables and setting breakpoints at the COBOL verbs under XPEDITER/TSO.

Compuware supplies a separate panel, **XPPISPFT**, which is functionally similar to the ISPF Dialog Test option 1. To use this panel to identify the dialog function to be debugged, enter **XPPISPFT** in the ISPF Menu field.

OPT
If you use XPPISPFT, leave this field blank. Otherwise, enter the starting point by ISPF option number (for example, 7.1).

APPLID
Enter the application ID.

File List/JCL Member
Enter the dataset name that contains the file list, CLIST, or JCL. The File Allocation Utility (FAU) preallocates files and databases that are processed by the program upon entry to the debugging session.

If the member name of a PDS is omitted, a member list is displayed.

If the dataset contains a CLIST, XPEDITER/TSO immediately executes it and begins the debugging session.

If the dataset contains a file list or JCL, the FAU is invoked to dynamically allocate the specified files before beginning the debugging session. Refer to Appendix A, "Using the File Allocation Utility" on page A-1 for detailed information about the FAU.

Code Coverage Test?
Enter **YES** if XPEDITER/Code Coverage data should be collected for this test. The default value is NO.

Note: The Code Coverage option is only available if you have purchased XPEDITER/Code Coverage.

Is This a DB2 Test?
Enter **YES** if the program executes EXEC SQL statements. XPEDITER/TSO selects the proper DB2 libraries for inclusion in the setup library (XTASKLIB) concatenation.

The DB2 connection must be established, such as executing the TSO DSN RUN command, before initiating the debugging session.

System
Enter the DB2 subsystem name. Use the subsystem name assigned to the DB2 DSNLOAD library to be included in XTASKLIB.

XPEDITER/DevEnterprise?
Enter **YES** in this field if you want to connect to XPEDITER/DevEnterprise. The default is NO.

Qualified LU name
Enter the fully qualified LU name if you are going to use XPEDITER/DevEnterprise. Enter the same name that was entered on the installation screen.

When you press **Enter** from the Dialog screen, the specified ISPF screen is displayed, but you must invoke the program or edit macro as you would if you were debugging without XPEDITER/TSO.

To invoke a function on the XPPISPFT screen, shown in Figure B-5 on page B-8, only one entry is required in one of the following fields: PANEL, CMD, or PGM.

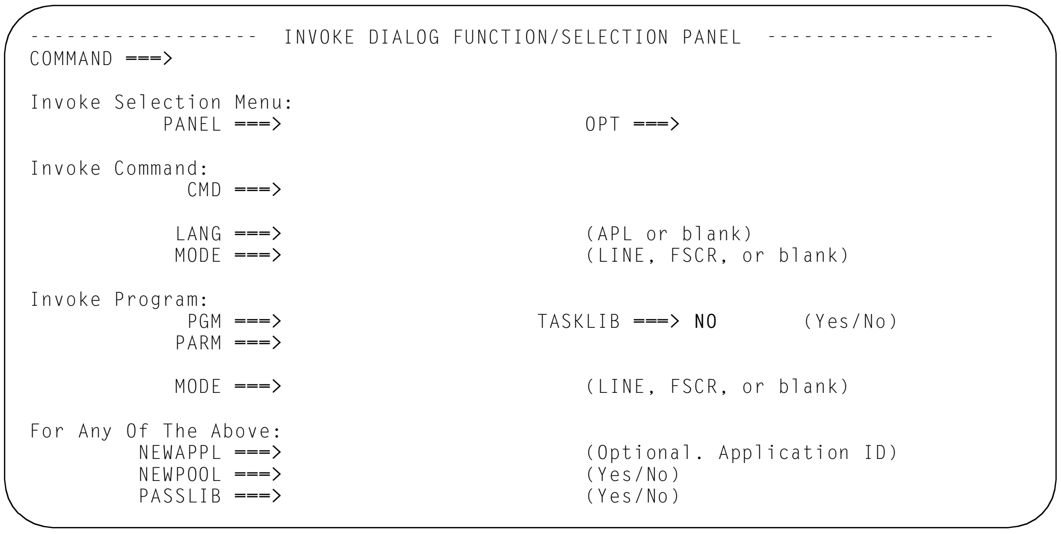
Note: When control is passed from XPEDITER/TSO to any user panel, all XPEDITER commands become inactive.

Effect of Dialog Program Intercepts

Each of the programs named on the Dialog screen must be an MVS load module name. Programs named in the intercept list must not be referenced within the XPEDITER session with the SOURCE or INTERCEPT command. These programs will be intercepted under XPEDITER dialog control when activated by sequences of LOAD/CALL, LINK, or ATTACH. Typical external events that can result in one of these sequences are: running the programs as TSO command processors, naming the programs as the target of a TSO CALL command, or executing the programs via ISPF SELECT services. Each of the programs should be independent of any other program named in the intercept list.

When the programs are not independent, special processing is performed by XPEDITER and user interaction is necessary to test multiple programs simultaneously. For example, assume that program PGMA calls program PGMB, which then calls program PGMC. Also, assume that all three programs are named as dialog intercepts. When PGMA is activated by the user dialog, an XPEDITER session is initiated and the source of PGMA is displayed. During the execution of PGMA, PGMB is eventually activated. However, since XPEDITER is already active for the PGMA intercept, PGMB is not intercepted and executes as if no intercept had been set. The same is true for PGMC.

There are two ways in which such a nested sequence of programs can be tested. The first method is to name only PGMA on the Dialog screen. When PGMA is intercepted, then SOURCE or INTERCEPT commands can be used to reference PGMB and PGMC. A second method can be used when programs are executed multiple times within the dialog. The START and MAX values determine when the intercepts are to be active for each program in the list. For example, let PGMA and PGMB both be named in the intercept list with the MAX value for PGMA set to 3. Now when the user dialog invokes PGMA repetitively, PGMA is intercepted each of the first three times it executes, but on the fourth and subsequent executions, when PGMA calls PGMB, XPEDITER intercepts PGMB.



```
------------------  INVOKE DIALOG FUNCTION/SELECTION PANEL  ------------------
COMMAND ===>

Invoke Selection Menu:
           PANEL ===>                        OPT ===>

Invoke Command:
             CMD ===>

            LANG ===>                        (APL or blank)
            MODE ===>                        (LINE, FSCR, or blank)

Invoke Program:
             PGM ===>                    TASKLIB ===> NO       (Yes/No)
            PARM ===>

            MODE ===>                        (LINE, FSCR, or blank)

For Any Of The Above:
         NEWAPPL ===>                        (Optional. Application ID)
         NEWPOOL ===>                        (Yes/No)
         PASSLIB ===>                        (Yes/No)
```

Figure B-5. XPPISPFT - Invoke Dialog Function/Selection Panel

The fields on the Invoke Dialog Function/Selection Panel are described below:

PANEL
Name of the ISPF panel to be invoked. **Required if CMD or PGM is not specified.**

OPT
Optional; the initial option that must be valid on the panel named above.

CMD
The name of a command procedure (CLIST or EXEC), or any TSO command, to be invoked as a dialog function. Command parameters can be included. **Required if PANEL or PGM is not specified.**

LANG
Specify **APL** if the string specified by the CMD keyword is an APL function and the APL2 environment is active. Otherwise, leave blank.

MODE
Determines whether line mode will be entered or not. LINE causes line mode to be entered. FSCR causes full screen mode to be entered. If you leave MODE blank when selecting a command procedure, line mode is entered unless you prefix the command with a percent sign (%).

PGM
The name of a program to be invoked as a dialog function. The program name must be one of the program names specified in the previous Dialog screen. Otherwise, it causes an abend. **Required if PANEL or CMD is not specified.**

PARM
Optional; parameters to be passed to the program.

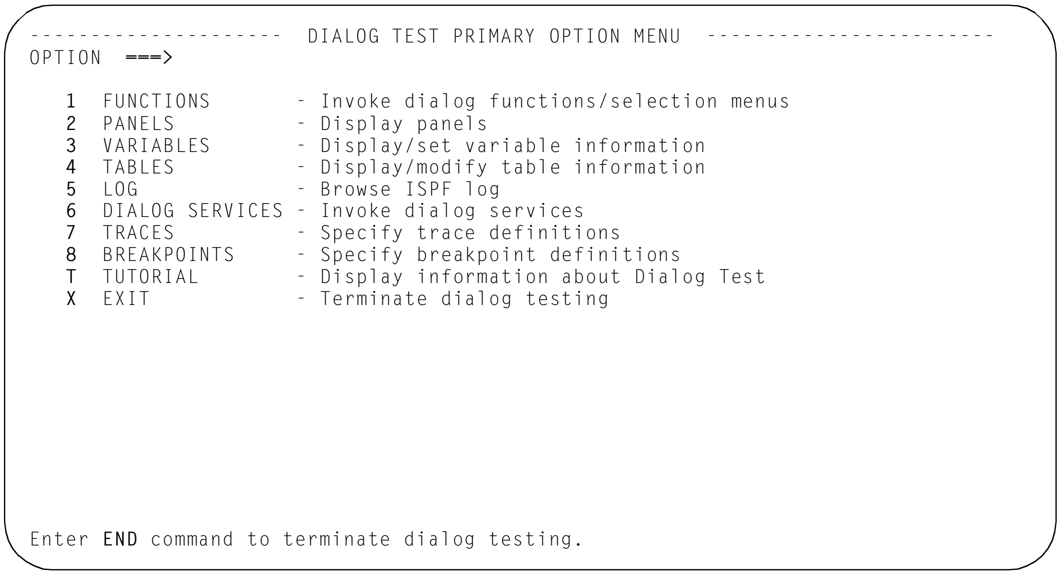
TASKLIB
Enter **Yes** if XTASKLIB is to be searched for the specified program name. The default is No.

NEWAPPL
Specifies the new application ID name, if one is being invoked. If a new application is specified, the next selection menu displayed is the application's primary option menu.

NEWPOOL
Enter **Yes** if a new shared variable pool is to be created. This value is ignored if NEWAPPL is specified.

PASSLIB
Enter **Yes** if the current set of application level ISPF libraries (if any exist) are to be passed to the application being selected. This field is valid only if NEWAPPL is specified.

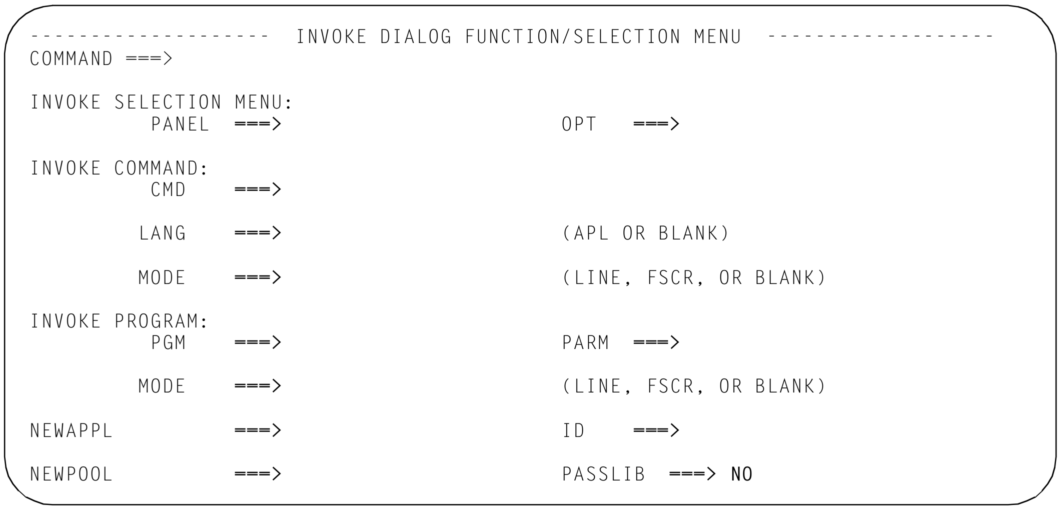
To invoke a function through the ISPF Menu ISR@PRIM, select option **1** (FUNCTIONS) on the Dialog Test Primary Option Menu shown in Figure B-6.



```
--------------------  DIALOG TEST PRIMARY OPTION MENU  -----------------------
OPTION  ===>

   1  FUNCTIONS       - Invoke dialog functions/selection menus
   2  PANELS          - Display panels
   3  VARIABLES       - Display/set variable information
   4  TABLES          - Display/modify table information
   5  LOG             - Browse ISPF log
   6  DIALOG SERVICES - Invoke dialog services
   7  TRACES          - Specify trace definitions
   8  BREAKPOINTS     - Specify breakpoint definitions
   T  TUTORIAL        - Display information about Dialog Test
   X  EXIT            - Terminate dialog testing









Enter END command to terminate dialog testing.
```

Figure B-6. Dialog Test Primary Option Menu

Then specify the function to be invoked in one of the following fields on the Invoke Dialog Function/Selection Menu shown in Figure B-7 on page B-10: PANEL, CMD, or PGM.



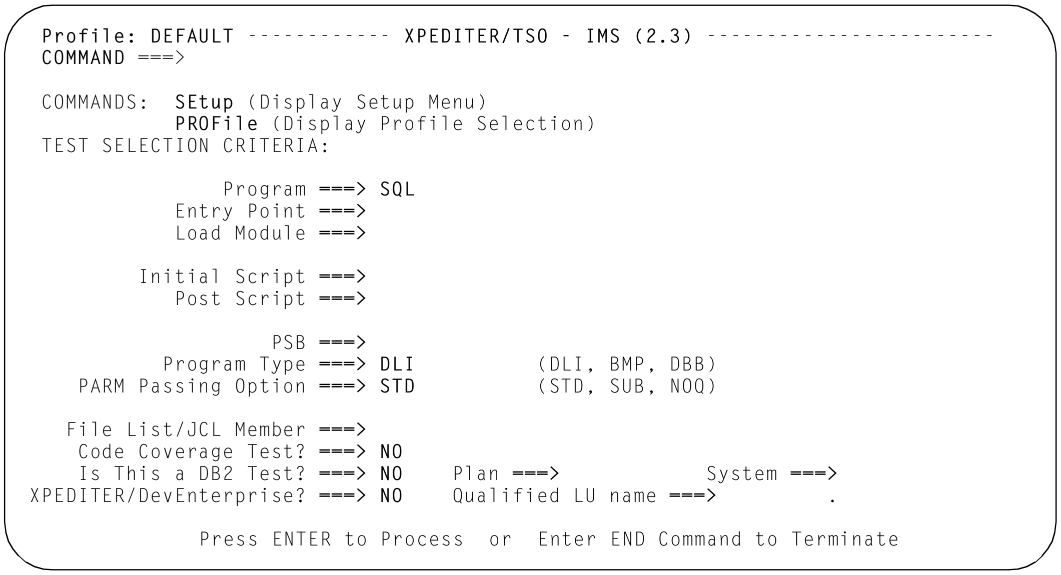
```
--------------------  INVOKE DIALOG FUNCTION/SELECTION MENU  ------------------
COMMAND ===>

INVOKE SELECTION MENU:
          PANEL  ===>                        OPT   ===>

INVOKE COMMAND:
          CMD    ===>

         LANG    ===>                        (APL OR BLANK)

         MODE    ===>                        (LINE, FSCR, OR BLANK)

INVOKE PROGRAM:
          PGM    ===>                        PARM  ===>

         MODE    ===>                        (LINE, FSCR, OR BLANK)

NEWAPPL          ===>                        ID    ===>

NEWPOOL          ===>                        PASSLIB  ===> NO
```

Figure B-7. ISR@PRIM - Invoke Dialog Function/Selection Menu

Press Enter to initiate the debugging session. Refer to Chapter 5, "Debugging Interactively" on page 5-1 for information on how to debug your program.

IMS Test Screen – Environments Menu - Option 3

The IMS Test screen, shown in Figure B-8, lets you specify the IMS/DB environment parameters and initiate a debugging session. Batch programs that issue IMS/DB database calls or both IMS/DB and DB2 database calls can be executed in this environment.



```
 Profile: DEFAULT ------------ XPEDITER/TSO - IMS (2.3) -----------------------
 COMMAND ===>

 COMMANDS:  SEtup (Display Setup Menu)
            PROFile (Display Profile Selection)
 TEST SELECTION CRITERIA:

                Program ===> SQL
            Entry Point ===>
            Load Module ===>

         Initial Script ===>
            Post Script ===>

                    PSB ===>
           Program Type ===> DLI          (DLI, BMP, DBB)
    PARM Passing Option ===> STD          (STD, SUB, NOQ)

   File List/JCL Member ===>
    Code Coverage Test? ===> NO
    Is This a DB2 Test? ===> NO    Plan ===>            System ===>
XPEDITER/DevEnterprise? ===> NO    Qualified LU name ===>         .

              Press ENTER to Process  or  Enter END Command to Terminate
```

Figure B-8. IMS Test Screen

The fields on the IMS Test screen are described below:

Program
: **This field is required**. Enter the name of the load module.

Note: If executing other than the linked entry name, then the program name is the CSECT name and the load module name must be specified under Load Module.

Entry Point

Enter an alternate entry point if execution is to start at a point other than the link-edited entry point.

Load Module

Enter only if executing a CSECT other than the link-edited entry point of a load module, such as unit testing CSECTs within a load module.

Initial Script

Enter the member name of a test script if you want to execute a predefined command stream at the *beginning* of the debugging session. This member will be executed after the inclusion of the Site-wide script member @@SITE@@, if defined.

Post Script

Enter the member name of a test script if you want to execute a predefined command stream at the *end* of the debugging session.

PSB

Required. Enter the name of the program specification block (PSB) associated with the program. The PSB defines the database accessed by the program and the type of I/O operations performed.The PSB name for MPP programs should be the same as the load module name.

If you are debugging a subordinate module that does not have a PSB, but does make DLI calls, you must supply the PSB of the main program.The USE command is used to specify which PCB in the PSB of the calling module should be associated with the PCB defined in the Linkage Section of the subroutine.

Program Type

Required. Enter the type of IMS program. It is generally the first parameter value passed to IMS in the JCL. This field can have one of the following values:

BMP	A BMP program is attached to a control region. The particular control region is determined by the IMS RESLIBS allocated and the IMSID within the BMP parameter.
DBB	A DBB program is processed in an offline DL/I batch processing region using ACB libraries instead of PSB and DBD libraries.
DLI	A DL/I program is processed in an offline DL/I batch processing region using PSB and DBD libraries.

Note: These program types are retrieved from the PARM lists in the Setup option. Your specific site options determine which program types are available on the screen.

PARM Passing Option

Required. Default values are prefilled for this field. The valid PARM passing options are:

STD	Specify **STD** when debugging an IMS program. XPEDITER/TSO passes the PARM string to the program unaltered. STD is the default.
SUB	Specify **SUB** when an IMS subroutine is to be debugged as a stand-alone program. No PARM string is needed because you are debugging an IMS subroutine without the IMS driver module present.Issue a USE command to assign addressability to the PSB at the beginning of the debugging session.
NOQ	Specify **NOQ** (no queue) when you are debugging an IMS/DC program with XPEDITER/TSO. XPEDITER/TSO matches the PARM string that IMS passes to theprogram with the PARM string that the program is expecting in reverse order. This lets you debug the logic and database calls of an IMS/DC program under TSO when a message queue is not available.

File List/JCL Member

Enter the dataset name that contains the file list, CLIST, or JCL. The File Allocation Utility (FAU) preallocates files and databases that are processed by the program upon entry to the debugging session.

If the member name of a PDS is omitted, a member list is displayed.

If the dataset contains a CLIST, XPEDITER/TSO immediately executes it and begins the debugging session.

If the dataset contains a file list or JCL, the FAU is invoked to dynamically allocate the specified files before beginning the debugging session. Refer to Appendix A, Using the File Allocation Utility" on page A-1 for detailed information about the FAU.

Code Coverage Test?

Enter **YES** if XPEDITER/Code Coverage data should be collected for this test. The default value is NO.

Note: The Code Coverage option is only available if you have purchased XPEDITER/Code Coverage.

Is This a DB2 Test?

Enter **YES** if the program executes EXEC SQL statements. The default value is NO.

Plan

Enter the DB2 plan name generated during the bind process if the program executes EXEC SQL statements. If omitted, the plan name defaults to the name specified in the Program field.

System

Enter the DB2 subsystem name. The subsystem name depends on the release level of DB2 allocated to the DSNLOAD library specified in the Setup option.

XPEDITER/DevEnterprise?

Enter **YES** in this field if you want to connect to XPEDITER/DevEnterprise. The default is NO.

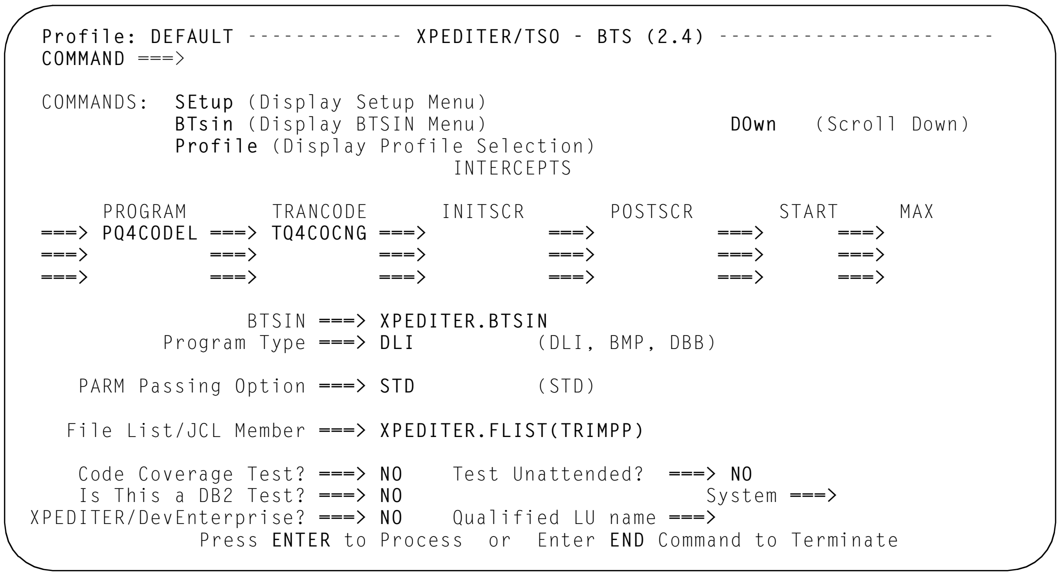
Qualified LU name

Enter the fully qualified LU name if you are going to use XPEDITER/DevEnterprise. Enter the same name that was entered on the installation screen.

Press **Enter** to initiate the debugging session. Refer to Chapter 5, "Debugging Interactively" on page 5-1 for information on how to debug your program.

BTS Test Screen – Environments Menu - Option 4

The BTS screen, shown in Figure B-9, lets you specify the BTS environment parameters and initiate a debugging session. IMS/DC, MPP, and BMP programs can be debugged with the use of BTS in this environment.



```
 Profile: DEFAULT ------------ XPEDITER/TSO - BTS (2.4) -----------------------
 COMMAND ===>

 COMMANDS:  SEtup (Display Setup Menu)
            BTsin (Display BTSIN Menu)                    DOwn   (Scroll Down)
            Profile (Display Profile Selection)
                                 INTERCEPTS

      PROGRAM       TRANCODE      INITSCR       POSTSCR       START     MAX
 ===> PQ4CODEL ===> TQ4COCNG ===>          ===>          ===>      ===>
 ===>          ===>          ===>          ===>          ===>      ===>
 ===>          ===>          ===>          ===>          ===>      ===>

                BTSIN ===> XPEDITER.BTSIN
          Program Type ===> DLI          (DLI, BMP, DBB)

    PARM Passing Option ===> STD          (STD)

   File List/JCL Member ===> XPEDITER.FLIST(TRIMPP)

    Code Coverage Test? ===> NO    Test Unattended?  ===> NO
    Is This a DB2 Test? ===> NO                        System ===>
XPEDITER/DevEnterprise? ===> NO    Qualified LU name ===>
              Press ENTER to Process  or  Enter END Command to Terminate
```

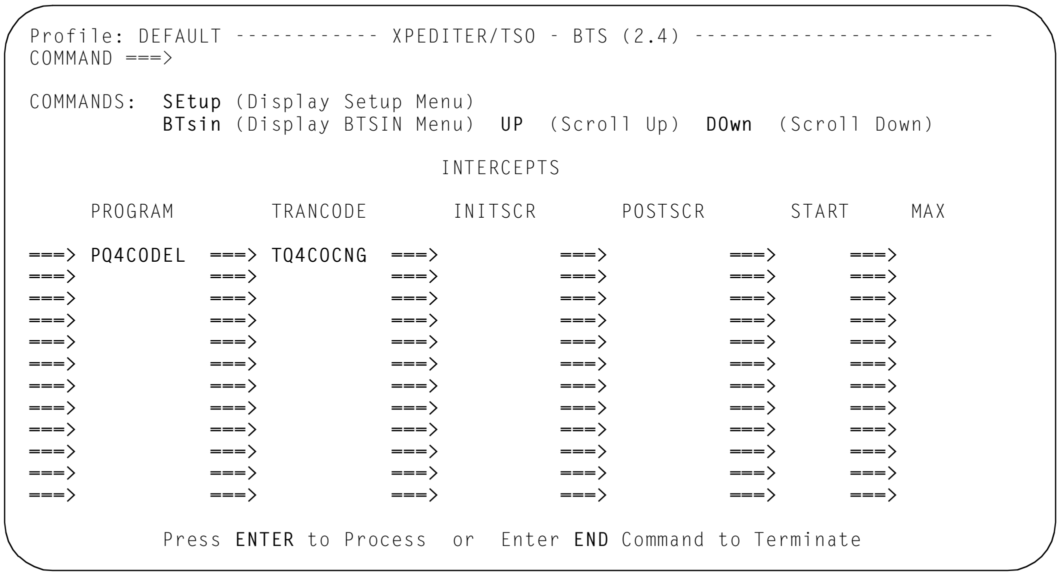
Figure B-9. BTS Screen

The fields on this screen are described below:

PROGRAM

Required if TRANCODE is not specified. Used to identify which programs are to be debugged by XPEDITER/TSO. You can enter the load module name in this field, the transaction code in the TRANCODE field, or supply values for both fields. If you enter only the program name, all the transactions associated with that program are intercepted by XPEDITER/TSO. If you enter only a transaction code, its associated program is intercepted.

Three programs or transaction codes can be specified on the BTS screen. If you want to continue the list, you can scroll down to a second screen, shown in Figure B-10 on page B-14. The last entry on the previous page is displayed in the top row. Once the second screen is full, you can scroll down to a third screen.Notice that on the second (and subsequent screens), the UP command is available in addition to the DOWN command.



```
 Profile: DEFAULT ----------- XPEDITER/TSO - BTS (2.4) ------------------------
 COMMAND ===>

 COMMANDS:  SEtup (Display Setup Menu)
            BTsin (Display BTSIN Menu)  UP  (Scroll Up)  DOwn  (Scroll Down)

                                  INTERCEPTS

      PROGRAM        TRANCODE       INITSCR       POSTSCR       START     MAX

 ===> PQ4CODEL  ===> TQ4COCNG  ===>          ===>          ===>      ===>
 ===>           ===>           ===>          ===>          ===>      ===>
 ===>           ===>           ===>          ===>          ===>      ===>
 ===>           ===>           ===>          ===>          ===>      ===>
 ===>           ===>           ===>          ===>          ===>      ===>
 ===>           ===>           ===>          ===>          ===>      ===>
 ===>           ===>           ===>          ===>          ===>      ===>
 ===>           ===>           ===>          ===>          ===>      ===>
 ===>           ===>           ===>          ===>          ===>      ===>
 ===>           ===>           ===>          ===>          ===>      ===>
 ===>           ===>           ===>          ===>          ===>      ===>
 ===>           ===>           ===>          ===>          ===>      ===>

            Press ENTER to Process  or  Enter END Command to Terminate
```

Figure B-10. BTS Screen 2

TRANCODE

Required if PROGRAM is not specified. Used to identify which transactions are to be debugged by XPEDITER/TSO. You can enter the transaction code in this field, the load module name in the PROGRAM field, or enter values for both fields.

Each transaction code must be unique. If you enter only a transaction code, its associated program is intercepted. However, multiple transactions can invoke the same program. Therefore, if you enter only the program name, all transactions associated with the program are intercepted for debugging under XPEDITER/TSO.

INITSCR

Enter the member name of a test script if you want to execute a predefined command stream at the *beginning* of the debugging session. This member will be executed after the inclusion of the Site-side script member @@SITE@@, if defined.

POSTSCR

Enter the member name of a test script if you want to execute a predefined command stream at the *end* of the debugging session.

START

Enter up to four digits to specify on which occurrence of the program invocation the intercept is to begin. By default, program intercept begins on the first occurrence. When the program is intercepted, the debugging session is initiated and the source is displayed.

MAX

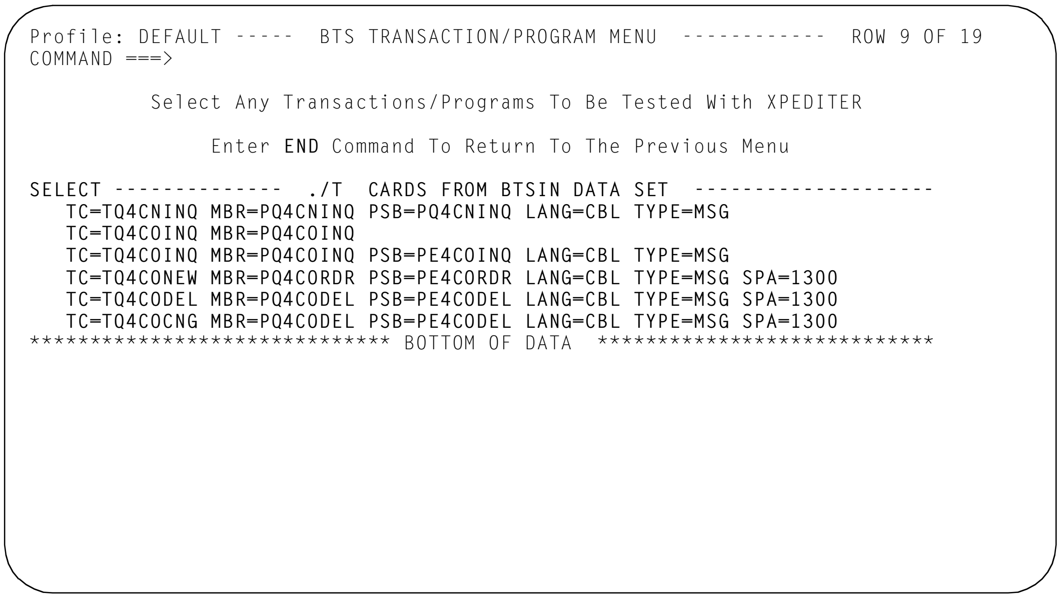
Enter up to four digits to specify the maximum number of times the program intercept is processed. If this field is left blank, the value defaults to an infinite number. If you enter an EXIT command and there are still intercepts remaining, those intercepts are ignored.

BTSIN

Required. XPEDITER/TSO scans the BTSIN dataset and searches for the program and transaction code to be debugged. An intercept is established for each one found. The BTSIN dataset must be either a sequential dataset or a member of a partitioned dataset. The BTSIN cards should not be modified.

Entering BTSIN on the command line of the BTS screendisplays a list of transactions (./T cards) in the BTSIN file (shown in Figure B-11). You can enter the **S** (Select) line command next to the transaction you want to debug and return to the BTS screen. The selected transactions are automatically filled in for the appropriate fields.

Note: When testing DB2 under the BTS/DLI environment, you must define the DB2 subsystem name to BTS by using the ./P statement as part of the BTSIN control cards.



```
Profile: DEFAULT -----  BTS TRANSACTION/PROGRAM MENU  -----------  ROW 9 OF 19
COMMAND ===>

          Select Any Transactions/Programs To Be Tested With XPEDITER

               Enter END Command To Return To The Previous Menu

SELECT --------------  ./T  CARDS FROM BTSIN DATA SET  --------------------
   TC=TQ4CNINQ MBR=PQ4CNINQ PSB=PQ4CNINQ LANG=CBL TYPE=MSG
   TC=TQ4COINQ MBR=PQ4COINQ
   TC=TQ4COINQ MBR=PQ4COINQ PSB=PE4COINQ LANG=CBL TYPE=MSG
   TC=TQ4CONEW MBR=PQ4CORDR PSB=PE4CORDR LANG=CBL TYPE=MSG SPA=1300
   TC=TQ4CODEL MBR=PQ4CODEL PSB=PE4CODEL LANG=CBL TYPE=MSG SPA=1300
   TC=TQ4COCNG MBR=PQ4CODEL PSB=PE4CODEL LANG=CBL TYPE=MSG SPA=1300
***************************** BOTTOM OF DATA  ***************************
```

Figure B-11. BTS Transaction/Program Menu

Program Type

Required. Enter the type of IMS program. It is generally the first parameter value passed to IMS in the JCL. This field can have one of the following values:

BMP	A BMP program is manually scheduled by the system operator and, typically, processes some data that was held by IMS in a message queue.
DBB	A DBB program is processed in an offline DL/I batch processing region using ACB libraries instead of PSB and DBD libraries.
DLI	A DL/I program is processed in an offline DL/I batch processing region using PSB and DBD libraries.

Note: These program types are retrieved from PARM lists in the Setup option. Your specific site options determine which program types are available on the screen.

PARM Passing Option

Required. Default values are prefilled for this field. The valid PARM passing option is STD. If you specify STD when debugging an IMS program, XPEDITER/TSO passes the PARM string to the program unaltered.

File List/JCL Member

Enter the dataset name that contains the file list, CLIST, or JCL. The File Allocation Utility (FAU) preallocates files and databases that are processed by the program upon entry to the debugging session.

If the member name of a PDS is omitted, a member list is displayed.

If the dataset contains a CLIST, XPEDITER/TSO immediately executes it and begins the debugging session.

If the dataset contains a file list or JCL, the FAU is invoked to dynamically allocate the specified files before beginning the debugging session. Refer to Appendix A, "Using the File Allocation Utility" on page A-1 for detailed information about the FAU.

Code Coverage Test?

Enter **YES** if XPEDITER/Code Coverage data should be collected for this test. The default value is NO.

Note: The Code Coverage option is only available if you have purchased XPEDITER/Code Coverage.

Test Unattended?
Enter **YES** in the 'Test Unattended?' field to run in Unattended mode. The default value is NO. In unattended mode, after BTS gives control to the application program, only XPEDITER commands are processed in the Initial Script, the Post Script, and the Abend Script. Interactive XPEDITER commands are not permitted (from the terminal).

Is This a DB2 Test?
Enter **YES** if the program executes EXEC SQL statements. The default value is NO.

System
Enter the DB2 subsystem name. The subsystem name depends on the release level of DB2 allocated to the DSNLOAD library specified in the Setup option.

Note: When testing DB2 under the BTS/DLI environment, you must define the DB2 subsystem name to BTS by using the ./P statement as part of the BTSIN control cards.

XPEDITER/DevEnterprise?
Enter **YES** in this field if you want to connect to XPEDITER/DevEnterprise. The default is NO.

Qualified LU name
Enter the fully qualified LU name if you are going to use XPEDITER/DevEnterprise. Enter the same name that was entered on the installation screen.

Press **Enter** to initiate the debugging session. BTS will be invoked first and prompt you for a transaction code to start a transaction. Refer to Chapter 5, "Debugging Interactively" on page 5-1 for information on how to debug your program.

Hogan BATCHPEM Test Screen – Environments Menu - Option 5

The Hogan BATCHPEM screen, shown in Figure B-12, lets you specify the Hogan BATCHPEM environment parameters and initiate a debugging session. Batch programs that run under the Hogan umbrella and process QSAM and VSAM files can be executed in this environment.

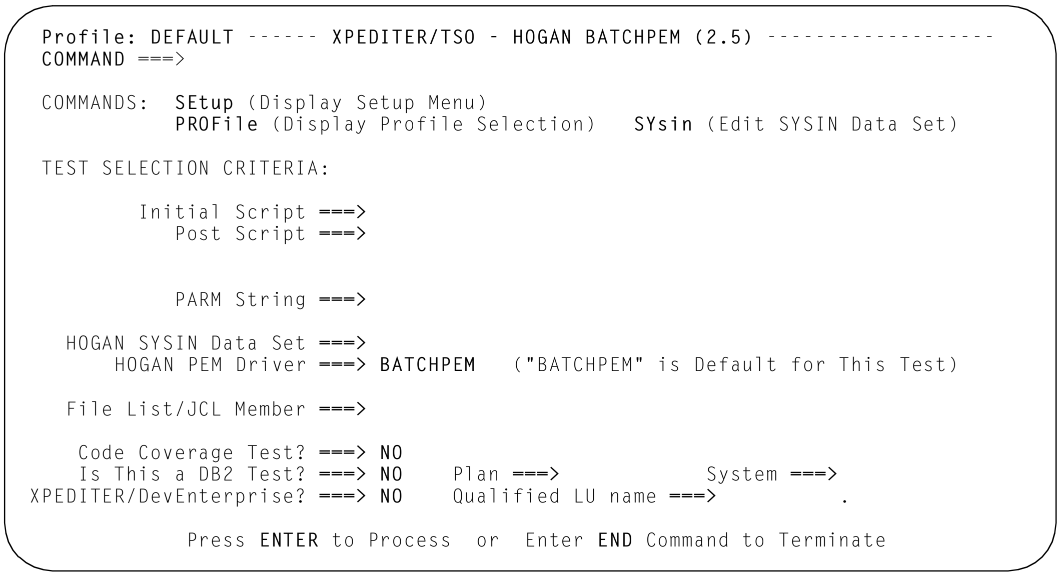

```
 Profile: DEFAULT ------ XPEDITER/TSO - HOGAN BATCHPEM (2.5) -------------------
 COMMAND ===>

 COMMANDS:  SEtup (Display Setup Menu)
            PROFile (Display Profile Selection)   SYsin (Edit SYSIN Data Set)

 TEST SELECTION CRITERIA:

         Initial Script ===>
            Post Script ===>


            PARM String ===>

   HOGAN SYSIN Data Set ===>
       HOGAN PEM Driver ===> BATCHPEM   ("BATCHPEM" is Default for This Test)

   File List/JCL Member ===>

    Code Coverage Test? ===> NO
    Is This a DB2 Test? ===> NO    Plan ===>            System ===>
XPEDITER/DevEnterprise? ===> NO    Qualified LU name ===>          .

             Press ENTER to Process  or  Enter END Command to Terminate
```

Figure B-12. Hogan BATCHPEM Screen

The fields on the Hogan BATCHPEM screen are defined below:

Initial Script

Enter the member name of a test script if you want to execute a predefined command stream at the *beginning* of the debugging session. This member will be executed after the inclusion of the Site-wide script member @@SITE@@, if defined.

Post Script

Enter the member name of a test script if you want to execute a predefined command stream at the *end* of the debugging session.

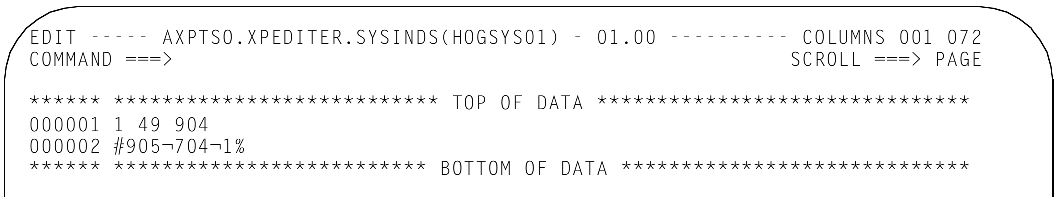
PARM String

Optional field. Enter the parameter string to be passed to the Hogan PEM driver program. You can enter up to 100 characters in this field.

If the run-time parameter consists of several substrings separated by commas, or if it contains special characters (/, =, etc.), enclose the entire parameter in quotes (single or double).

Hogan SYSIN Data Set

A control file used by Hogan programs. Enter **SYSIN** on the command line to edit or create the SYSIN dataset, shown in Figure B-13.



```
EDIT ----- AXPTSO.XPEDITER.SYSINDS(HOGSYS01) - 01.00 ---------- COLUMNS 001 072
COMMAND ===>                                                    SCROLL ===> PAGE

****** *************************** TOP OF DATA ******************************
000001 1 49 904
000002 #905¬704¬1%
****** ************************** BOTTOM OF DATA ****************************
```

Figure B-13. Displaying the SYSIN Dataset

Hogan PEM Driver

The name of the Hogan driver. BATCHPEM is the default.

File List/JCL Member

Enter the dataset name that contains the file list, CLIST, or JCL. The File Allocation Utility (FAU) preallocates files and databases that are processed by the program upon entry to the debugging session. Any files or databases that are not dynamically allocated by Hogan must be allocated through this file.

If the member name of a PDS is omitted, a member list is displayed.

If the dataset contains a CLIST, XPEDITER/TSO immediately executes it and begins the debugging session.

If the dataset contains a file list or JCL, the FAU is invoked to dynamically allocate the specified files before beginning the debugging session. Refer to Appendix A, "Using the File Allocation Utility" on page A-1 for detailed information about the FAU.

Note: All CDMF files and databases should be allocated.

Code Coverage Test?

Enter **YES** if XPEDITER/Code Coverage data should be collected for this test. The default value is NO.

Note: The Code Coverage option is only available if you have purchased XPEDITER/Code Coverage.

Is This a DB2 Test?

Enter **YES** if the program executes EXEC SQL statements. The default value is NO.

Plan

Enter the DB2 plan name generated during the bind process if the program executes EXEC SQL statements.

System

Enter the DB2 subsystem name. The subsystem name depends on the release level of DB2 and is assigned at the time of installation.

XPEDITER/DevEnterprise?
Enter **YES** in this field if you want to connect to XPEDITER/DevEnterprise. The default is NO.

Qualified LU name
Enter the fully qualified LU name if you are going to use XPEDITER/DevEnterprise. Enter the same name that was entered on the installation screen.

Press **Enter** to initiate the debugging session. Refer to Chapter 5, "Debugging Interactively" on page 5-1 for information on how to debug your program.

Hogan DLIPEM Test Screen – Environments Menu - Option 6

The Hogan DLIPEM screen, shown in Figure B-14, lets you specify the Hogan DLIPEM environment parameters and initiate a debugging session. Batch programs that run under the Hogan umbrella and issue IMS/DB calls can be executed in this environment.

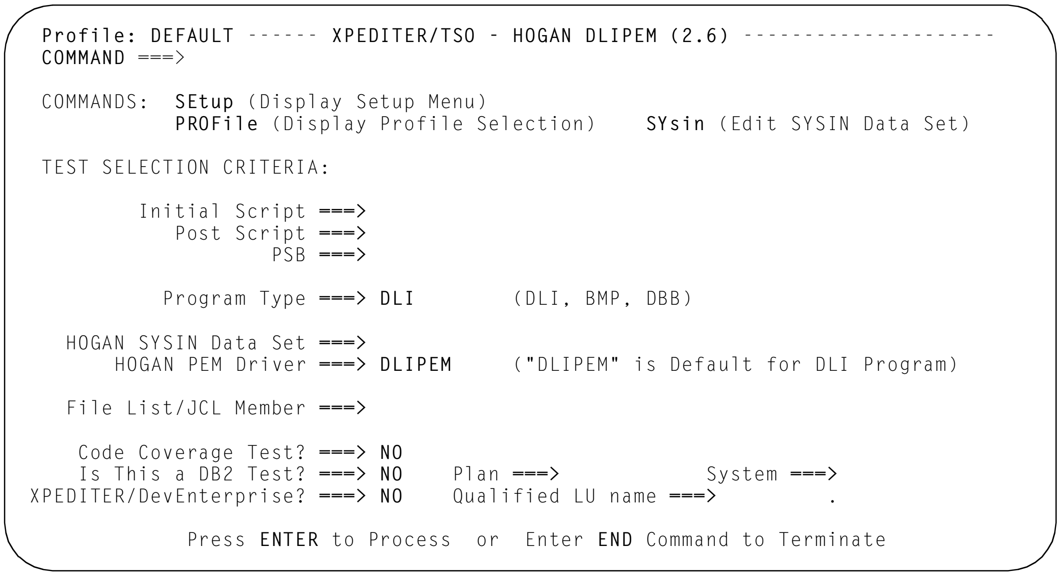

```
  Profile: DEFAULT ------ XPEDITER/TSO - HOGAN DLIPEM (2.6) --------------------
  COMMAND ===>

  COMMANDS:  SEtup (Display Setup Menu)
             PROFile (Display Profile Selection)    SYsin (Edit SYSIN Data Set)

  TEST SELECTION CRITERIA:

          Initial Script ===>
             Post Script ===>
                     PSB ===>

            Program Type ===> DLI        (DLI, BMP, DBB)

     HOGAN SYSIN Data Set ===>
         HOGAN PEM Driver ===> DLIPEM     ("DLIPEM" is Default for DLI Program)

     File List/JCL Member ===>

      Code Coverage Test? ===> NO
      Is This a DB2 Test? ===> NO    Plan ===>            System ===>
 XPEDITER/DevEnterprise? ===> NO    Qualified LU name ===>         .

             Press ENTER to Process  or  Enter END Command to Terminate
```

Figure B-14. Hogan DLIPEM Screen

The fields on the Hogan DLIPEM screen are:

Initial Script
Enter the member name of a test script if you want to execute a predefined command stream at the *beginning* of the debugging session. This member will be executed after the inclusion of the Site-wide script member @@SITE@@, if defined.

Post Script
Enter the member name of a test script if you want to execute a predefined command stream at the *end* of the debugging session.

PSB
The program specification block (PSB) associated with the program you are debugging.

Program Type
Required. Default values are prefilled for this field. Indicates the kind of IMS program you are debugging with IMS under XPEDITER/TSO. This field can have one of the following values:

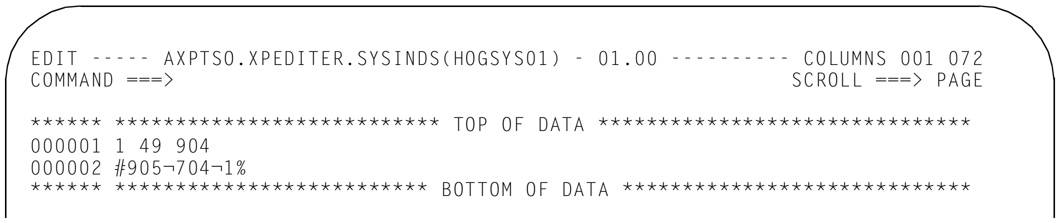
DLI A DL/I program is processed in an offline DL/I batch processing region using PSB and DBD libraries if your program is **not** connected to the IMS control region and does not use an ACB library,

BMP A BMP program is manually scheduled by the system operator and, typically, processes some data that was held by IMS in a message queue if your program is connected to the IMS control region.

DBB A DBB program is processed in an offline DL/I batch processing region using ACB libraries instead of PSB and DBD libraries if your program is **not** connected to the IMS control region and uses an ACB library.

Hogan SYSIN Data Set

A control file used by Hogan programs. Enter **SYSIN** on the command line to edit or create the SYSIN dataset, shown in Figure B-15.



```
EDIT ----- AXPTSO.XPEDITER.SYSINDS(HOGSYS01) - 01.00 --------- COLUMNS 001 072
COMMAND ===>                                                  SCROLL ===> PAGE

****** *************************** TOP OF DATA ******************************
000001 1 49 904
000002 #905¬704¬1%
****** ************************* BOTTOM OF DATA ****************************
```

Figure B-15. Displaying the SYSIN Dataset

Hogan PEM Driver

The name of the Hogan driver. DLIPEM is the default Hogan driver for DL/I or DBB programs. BMPPEM is the default for BMP programs.

File List/JCL Member

Enter the dataset name that contains the file list, CLIST, or JCL. The File Allocation Utility (FAU) preallocates files and databases that are processed by the program upon entry to the debugging session. Any files or databases that are not dynamically allocated by Hogan must be allocated through this file.

If the member name of a PDS is omitted, a member list is displayed.

If the dataset contains a CLIST, XPEDITER/TSO immediately executes it and begins the debugging session.

If the dataset contains a file list or JCL, the FAU is invoked to dynamically allocate the specified files before beginning the debugging session. Refer to Appendix A, "Using the File Allocation Facility" on page A-1 for detailed information about the FAU.

Code Coverage Test?

Enter **YES** if XPEDITER/Code Coverage data should be collected for this test. The default value is NO.

Note: The Code Coverage option is only available if you have purchased XPEDITER/Code Coverage.

Is This a DB2 Test?

Enter **YES** if the program executes EXEC SQL statements. The default value is NO.

Plan

Enter the DB2 plan name generated during the bind process if the program executes EXEC SQL statements. If omitted, the plan name defaults to the name specified in the Program field.

System

Enter the DB2 subsystem name. The subsystem name depends on the release level of DB2 and is assigned at the time of installation.

XPEDITER/DevEnterprise?

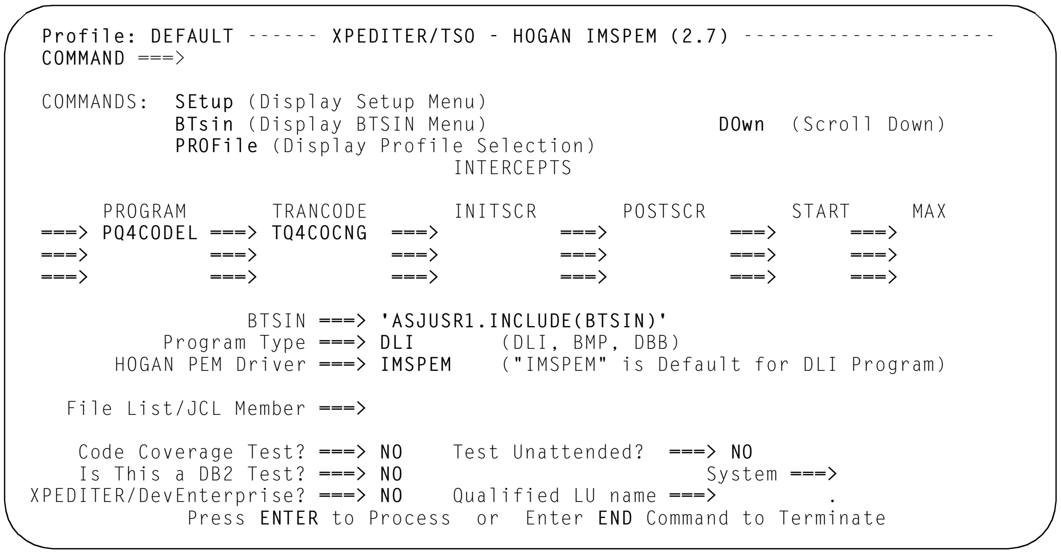
Enter **YES** in this field if you want to connect to XPEDITER/DevEnterprise. The default is NO

Qualified LU name
Enter the fully qualified LU name if you are going to use XPEDITER/DevEnterprise. Enter the same name that was entered on the installation screen.

Press **Enter** to initiate the debugging session. Refer to Chapter 5, "Debugging Interactively" on page 5-1 for information on how to debug your program.

Hogan IMSPEM Test Screen – Environments Menu - Option 7

The Hogan IMSPEM screen, displayed in Figure B-16, lets you specify debug parameters before beginning execution of a program in a Hogan IMSPEM environment.



```
  Profile: DEFAULT ------ XPEDITER/TSO - HOGAN IMSPEM (2.7) ---------------------
  COMMAND ===>

  COMMANDS:  SEtup (Display Setup Menu)
             BTsin (Display BTSIN Menu)                 DOwn  (Scroll Down)
             PROFile (Display Profile Selection)
                                   INTERCEPTS

       PROGRAM       TRANCODE       INITSCR       POSTSCR       START     MAX
  ===> PQ4CODEL ===> TQ4COCNG  ===>          ===>          ===>      ===>
  ===>          ===>           ===>          ===>          ===>      ===>
  ===>          ===>           ===>          ===>          ===>      ===>

                  BTSIN ===> 'ASJUSR1.INCLUDE(BTSIN)'
             Program Type ===> DLI       (DLI, BMP, DBB)
         HOGAN PEM Driver ===> IMSPEM    ("IMSPEM" is Default for DLI Program)

    File List/JCL Member ===>

     Code Coverage Test? ===> NO    Test Unattended?  ===> NO
     Is This a DB2 Test? ===> NO                         System ===>
 XPEDITER/DevEnterprise? ===> NO    Qualified LU name ===>         .
             Press ENTER to Process  or  Enter END Command to Terminate
```

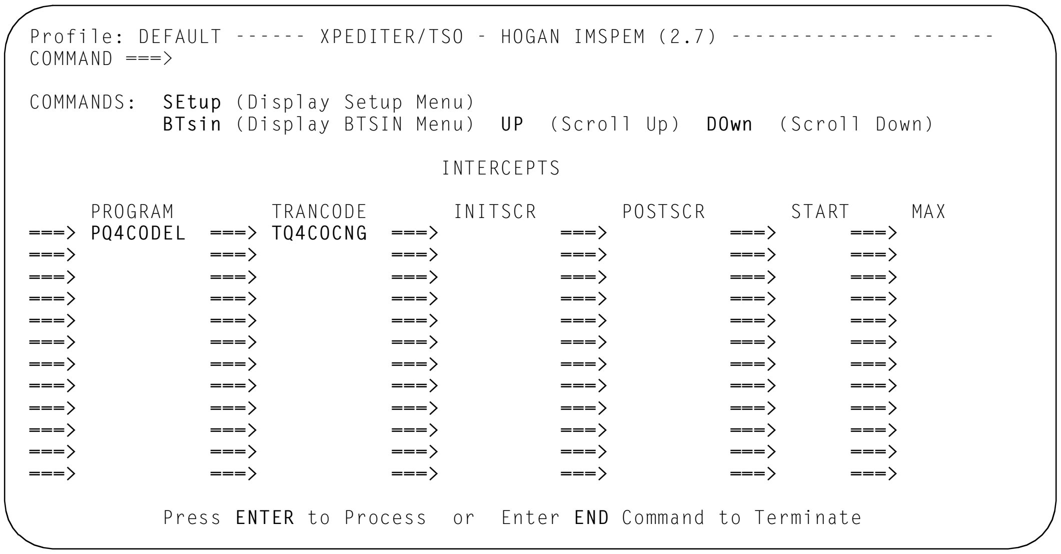
Figure B-16. Hogan IMSPEM Screen

The fields on the Hogan IMSPEM screen are:

PROGRAM
This field is required if TRANCODE is not specified. Used to identify which programs are to be debugged by XPEDITER/TSO. You can enter the load module name in this field, the transaction code in the TRANCODE field, or supply values for both fields. If you enter only the program name, all the transactions associated with that program are intercepted by XPEDITER/TSO. If you enter only a transaction code, its associated program is invoked.

Three programs or transaction codes can be specified on the Hogan IMSPEM screen. If you want to continue the list, you can scroll down to a second screen, shown in Figure B-17. The last entry on the previous page is displayed in the top row. Once the second screen is full, you can scroll down to a third screen. Notice that on the second (and subsequent screens), the UP command is available in addition to the DOWN command.

Note: XPEDITER/TSO processes every intercept specified on the panel and there is a memory overhead associated with each program specified.



```
Profile: DEFAULT ------ XPEDITER/TSO - HOGAN IMSPEM (2.7) ------------- -------
COMMAND ===>

COMMANDS:  SEtup (Display Setup Menu)
           BTsin (Display BTSIN Menu)  UP  (Scroll Up)  DOwn  (Scroll Down)

                                   INTERCEPTS

     PROGRAM        TRANCODE        INITSCR        POSTSCR        START     MAX
===> PQ4CODEL  ===> TQ4COCNG  ===>           ===>           ===>       ===>
===>           ===>           ===>           ===>           ===>       ===>
===>           ===>           ===>           ===>           ===>       ===>
===>           ===>           ===>           ===>           ===>       ===>
===>           ===>           ===>           ===>           ===>       ===>
===>           ===>           ===>           ===>           ===>       ===>
===>           ===>           ===>           ===>           ===>       ===>
===>           ===>           ===>           ===>           ===>       ===>
===>           ===>           ===>           ===>           ===>       ===>
===>           ===>           ===>           ===>           ===>       ===>
===>           ===>           ===>           ===>           ===>       ===>
===>           ===>           ===>           ===>           ===>       ===>

           Press ENTER to Process  or  Enter END Command to Terminate
```

Figure B-17. Hogan IMSPEM Screen 2

TRANCODE

Required if PROGRAM is not specified. Used to identify which transactions are to be debugged by XPEDITER/TSO. You can enter the transaction code in this field, the load module name in the PROGRAM field, or enter values for both fields.

Each transaction code must be unique. If you enter only a transaction code, its associated program is invoked. However, multiple transactions can invoke the same program. Therefore, if you enter only the program name, all the transactions associated with that program are intercepted and debugged by XPEDITER/TSO.

INITSCR

Enter the member name of a test script if you want to execute a predefined command stream at the *beginning* of the debugging session. This member will be executed after the inclusion of the Site-wide script member @@SITE@@, if defined.

POSTSCR

Enter the member name of a test script if you want to execute a predefined command stream at the *end* of the debugging session.

START

Enter up to four digits to specify on which occurrence of the program invocation the intercept is to begin. By default, program intercept begins on the first occurrence. When the program is intercepted, the debugging session is initiated and the source is displayed.

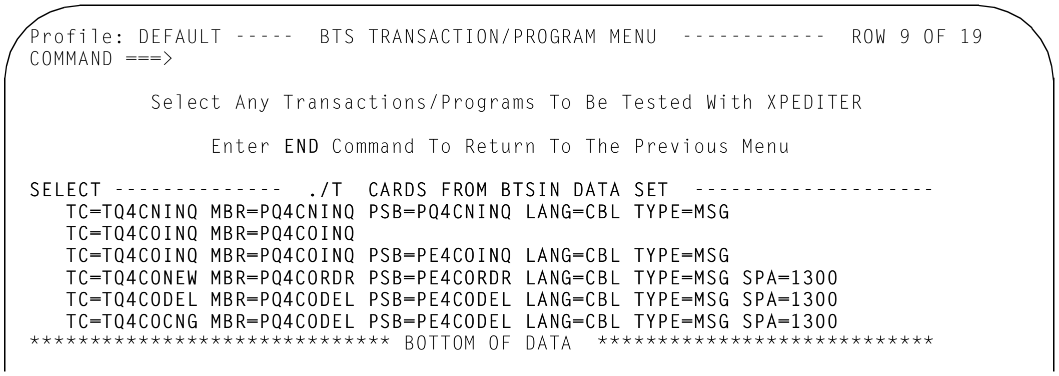
MAX

Enter up to four digits to specify the maximum number of times the program intercept is processed. If this field is left blank, the value defaults to an infinite number. If you enter **EXIT** and there are still intercepts remaining, those intercepts are ignored.

BTSIN

Required. XPEDITER/TSO scans the BTSIN dataset and searches for the program and transaction code to be debugged. An intercept is established for each one found. The BTSIN dataset must be either a sequential dataset or a member of a partitioned dataset. The BTSIN cards should not be modified.

Entering BTSIN on the command line of the HOGAN IMSPEM screen displays a list of transactions (./T cards) in the BTSIN file shown in B-18. You can enter the **S** (Select) line command next to the transaction you want to debug and return to the Hogan IMSPEM screen. The selected transactions are automatically filled in for the appropriate fields.



```
Profile: DEFAULT -----  BTS TRANSACTION/PROGRAM MENU  -----------  ROW 9 OF 19
COMMAND ===>

          Select Any Transactions/Programs To Be Tested With XPEDITER

               Enter END Command To Return To The Previous Menu

SELECT --------------  ./T  CARDS FROM BTSIN DATA SET  --------------------
   TC=TQ4CNINQ MBR=PQ4CNINQ PSB=PQ4CNINQ LANG=CBL TYPE=MSG
   TC=TQ4COINQ MBR=PQ4COINQ
   TC=TQ4COINQ MBR=PQ4COINQ PSB=PE4COINQ LANG=CBL TYPE=MSG
   TC=TQ4CONEW MBR=PQ4CORDR PSB=PE4CORDR LANG=CBL TYPE=MSG SPA=1300
   TC=TQ4CODEL MBR=PQ4CODEL PSB=PE4CODEL LANG=CBL TYPE=MSG SPA=1300
   TC=TQ4COCNG MBR=PQ4CODEL PSB=PE4CODEL LANG=CBL TYPE=MSG SPA=1300
****************************** BOTTOM OF DATA  ****************************
```

Figure B-18. BTS Transaction/Program Menu

Program Type

Required. Enter the type of IMS program. It is generally the first parameter value passed to IMS in the JCL. This field can have one of the following values:

MPP An MPP program is automatically loaded by IMS when a message to be processed by that program is received. The scheduling of MPP programs is entirely under the control of IMS.

BMP A BMP program is manually scheduled by the system operator and, typically, processes some data that was held by IMS in a message queue.

DBB A DBB program is processed in an offline DL/I batch processing region using ACB libraries instead of PDB and DBD libraries.

DLI A DL/I program is processed in an offline DL/I batch processing region using PSB and DBD libraries.

Note: These program types are retrieved from PARM lists in the Setup option. Your specific site options determine which program types are available on the screen.

Hogan PEM Driver

Enter the name of the Hogan PEM driver. IMSPEM is the default Hogan PEM driver for BTS testing.

File List/JCL Member

Enter the dataset name that contains the file list, CLIST, or JCL. The File Allocation Utility (FAU) preallocates files and databases that are processed by the program upon entry to the debugging session.

If the member name of a PDS is omitted, a member list is displayed.

If the dataset contains a CLIST, XPEDITER/IMS immediately executes it and begins the debugging session.

If the dataset contains a file list or JCL, the FAU is invoked to dynamically allocate the specified files before beginning the debugging session. Refer to Appendix A, "Using the File Allocation Utility" on page A-1 for detailed information about the FAU.

Note: To allocate a dataset to a BTS ddname, enter SETUP on the command line of the Hogan IMSPEM screen and then select option B on the Setup Menu. The BTS Test Setup Options screen is displayed. You can enter/verify the installed defaults for any BTS ddnames listed on the Setup Menu. Refer to "BTS Setup Menu" on page C-22 for more information.

Code Coverage Test?

Enter **YES** if XPEDITER/Code Coverage data should be collected for this test. The default value is NO.

Note: The Code Coverage option is only available if you have purchased XPEDITER/Code Coverage.

Test Unattended?
Enter **YES** in the 'Test Unattended?' field to run in Unattended mode. The default value is NO. In unattended mode, after BTS gives control to the application program, only XPEDITER commands are processed in the Initial Script, the Post Script, and the Abend Script. Interactive XPEDITER commands are not permitted (from the terminal).

Is This a DB2 Test?
Enter **YES** if the program executes EXEC SQL statements. The default value is NO.

System
Enter the DB2 subsystem name if the program executes EXEC SQL statements. The subsystem name depends on the release level of DB2 assigned to the DSNLOAD library specified at installation time.

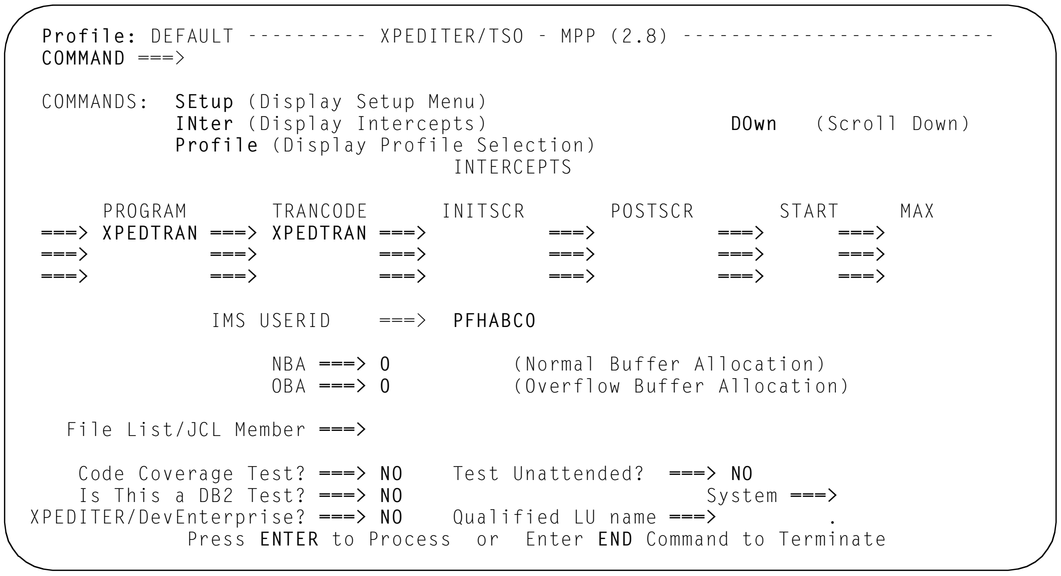
XPEDITER/DevEnterprise?
Enter **YES** in this field if you want to connect to XPEDITER/DevEnterprise. The default is NO.

Qualified LU name
Enter the fully qualified LU name if you are going to use XPEDITER/DevEnterprise. Enter the same name that was entered on the installation screen.

Press **Enter** to initiate the debugging session. Refer to Chapter 5, "Debugging Interactively" on page 5-1 for information on how to debug your program.

MPP Test Screen – Environments Menu - Option 8

The MPP Test screen displayed in Figure B-19 lets you set up environment parameters before beginning program execution of a program in an IMS message region. When you identify the transactions to be debugged and initiate a session, XPEDITER/IMS attaches the IMS message region within the TSO address space. You can enter the transaction code from an IMS terminal to start the transaction, and the source will be displayed on the TSO terminal where the debugging session was initiated. The operation of the product requires one logical TSO terminal and one logical IMS terminal (can be an ATM terminal), both on the same CPU.



```
 Profile: DEFAULT ---------- XPEDITER/TSO - MPP (2.8) -------------------------
 COMMAND ===>

 COMMANDS:  SEtup (Display Setup Menu)
            INter (Display Intercepts)                 DOwn   (Scroll Down)
            Profile (Display Profile Selection)
                                  INTERCEPTS

      PROGRAM       TRANCODE      INITSCR       POSTSCR       START     MAX
 ===> XPEDTRAN ===> XPEDTRAN ===>          ===>          ===>      ===>
 ===>          ===>          ===>          ===>          ===>      ===>
 ===>          ===>          ===>          ===>          ===>      ===>

              IMS USERID    ===>  PFHABC0

                   NBA ===> 0          (Normal Buffer Allocation)
                   OBA ===> 0          (Overflow Buffer Allocation)

   File List/JCL Member ===>

    Code Coverage Test? ===> NO    Test Unattended?  ===> NO
     Is This a DB2 Test? ===> NO                          System ===>
XPEDITER/DevEnterprise? ===> NO    Qualified LU name ===>         .
             Press ENTER to Process  or  Enter END Command to Terminate
```

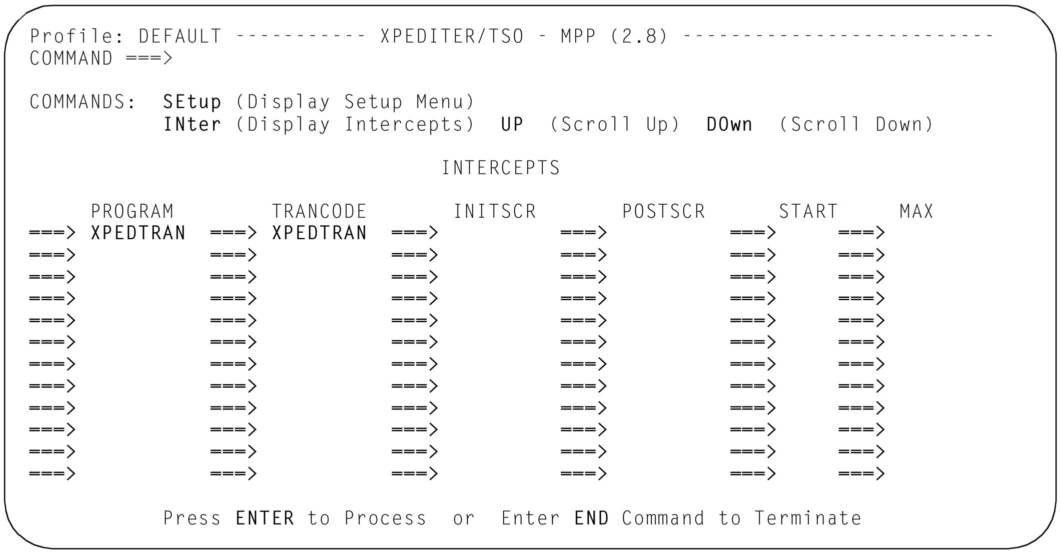
Figure B-19. MPP Test Screen

The fields on the MPP Test screen are:

PROGRAM

Required if TRANCODE is not specified. Used to identify which programs are to be debugged by XPEDITER/IMS. You can enter the load module name in this field, the transaction code in the TRANCODE field, or supply values for both fields. If you enter only the program name, all the transactions associated with that program will be intercepted by XPEDITER/IMS. If you enter only a transaction code, its associated program is intercepted.

Three programs or transaction codes can be specified on the MPP screen. If you want to continue the list, you can scroll down to a second MPP Test screen, shown in Figure B-20. The last entry on the previous page is displayed in the top row. Once the second screen is full, you can scroll down to a third screen. Notice that on the second (and subsequent screens), the UP command is available in addition to the DOWN command.



```
Profile: DEFAULT ---------- XPEDITER/TSO - MPP (2.8) ------------------------
COMMAND ===>

COMMANDS:  SEtup (Display Setup Menu)
           INter (Display Intercepts)  UP  (Scroll Up)  DOwn  (Scroll Down)

                                 INTERCEPTS

     PROGRAM        TRANCODE       INITSCR       POSTSCR      START     MAX
===> XPEDTRAN  ===> XPEDTRAN  ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>

           Press ENTER to Process  or  Enter END Command to Terminate
```

Figure B-20. MPP Test Screen 2

TRANCODE

Used to identify which transactions are to be debugged by XPEDITER/IMS. You can enter the transaction code in this field, the load module name in the PROGRAM field, or supply values for both fields.

This field is required if PROGRAM is not specified or if multiple transactions are associated with the program.

INITSCR

Enter the member name of a test script if you want to execute a predefined command stream at the *beginning* of the debugging session. This member will be executed after the inclusion of the Site-wide script member @@SITE@@, if defined.

POSTSCR

Enter the member name of a test script if you want to execute a predefined command stream at the *end* of the debugging session.

START

Enter up to four digits to specify on which occurrence of the program invocation the intercept is to begin. By default, program intercept begins on the first occurrence. When the program is intercepted, the debugging session is initiated and the source is displayed.

MAX

Enter up to four digits to specify the maximum number of times the program intercept is processed. If this field is left blank, the value defaults to an infinite number. If you enter an EXIT command and there are still intercepts remaining, those intercepts are ignored.

IMS USERID
Enter the XPEDITER/IMS USERID, up to eight digits.

NBA
Enter the Normal Buffer Allocation (NBA), up to two digits. The default value is 0.

OBA
Enter the Overflow Buffer Allocation (OBA), up to two digits. The default value is 0.

Note: The NBA and OBA fields appear on the screen only if your site installer indicated that your site uses Fast Path databases. The values are inserted in the PARM string when the IMS region is attached.

File List/JCL Member
Enter the dataset name that contains the file list, CLIST, or JCL. The File Allocation Utility (FAU) preallocates files and databases that are processed by the program upon entry to the debugging session.

If the member name of a PDS is omitted, a member list is displayed.

If the dataset contains a CLIST, XPEDITER/IMS immediately executes it and begins the debugging session.

If the dataset contains a file list or JCL, the FAU is invoked to dynamically allocate the specified files before beginning the debugging session. Refer to Appendix A, "Using the File Allocation Utility" on page A-1 for detailed information about the FAU.

Code Coverage Test?
Enter **YES** if XPEDITER/Code Coverage data should be collected for this test. The default value is NO.

Note: The Code Coverage option is only available if you have purchased XPEDITER/Code Coverage.

Test Unattended?
Enter **YES** in the 'Test Unattended?' field to run in Unattended mode. The default value is NO. In unattended mode, after the XPEDITER/IMS Message Processing Region (MPR) is attached, only XPEDITER commands are processed in the Initial Script, Post Script, and Abend Script. The TSO terminal remains locked and XPEDITER commands are not allowed from the terminal. When you are finished testing in the XPEDITER/IMS MPR, the MPR may be stopped and the TSO terminal unlocked by using either the XPEDITER Stop Region transaction, XPST, or the XPEDITER Stop Region BMP procedure XPSTOP. Refer to Chapter 7, "Stopping the XPEDITER/IMS Dependent Region" on page 7-12.

Is This a DB2 Test?
Enter **YES** if the program executes EXEC SQL statements. The default value is NO.

System
Enter the DB2 subsystem name if the program executes EXEC SQL statements. The subsystem name depends on the release level of DB2 and is assigned at the time of installation.

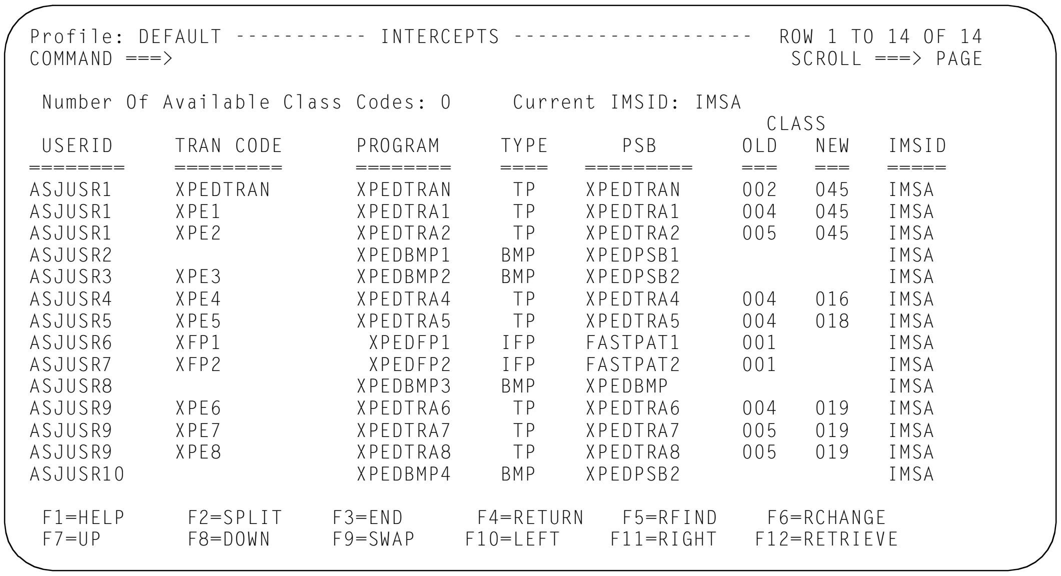
XPEDITER/DevEnterprise?
Enter **YES** in this field if you want to connect to XPEDITER/DevEnterprise. The default is NO.

Qualified LU name
Enter the fully qualified LU name if you are going to use XPEDITER/DevEnterprise. Enter the same name that was entered on the installation screen.

Press **Enter** to initiate the debugging session. Refer to Chapter 5, "Debugging Interactively" on page 5-1 for information on how to debug your program.

When intercepts cannot be set and the message `MAX USERS` is displayed on the test screen, enter INTER on the command line to look at the Intercepts screen, shown in Figure B-21 on page B-26. The Number of Available Class Codes field will be zero (0), indicating that the maximum number of users are testing with XPEDITER/IMS.

The Intercepts screen provides valuable information regarding the use of XPEDITER/IMS—how many users can still test using XPEDITER/IMS, what programs or transactions are being tested, and other information connected with the program and the transaction code.



```
Profile: DEFAULT ----------- INTERCEPTS --------------------  ROW 1 TO 14 OF 14
COMMAND ===>                                                   SCROLL ===> PAGE

 Number Of Available Class Codes: 0     Current IMSID: IMSA
                                                             CLASS
 USERID     TRAN CODE      PROGRAM     TYPE      PSB       OLD   NEW   IMSID
========    =========      ========    ====   =========    ===   ===   =====
ASJUSR1     XPEDTRAN       XPEDTRAN     TP    XPEDTRAN     002   045   IMSA
ASJUSR1     XPE1           XPEDTRA1     TP    XPEDTRA1     004   045   IMSA
ASJUSR1     XPE2           XPEDTRA2     TP    XPEDTRA2     005   045   IMSA
ASJUSR2                    XPEDBMP1    BMP    XPEDPSB1                 IMSA
ASJUSR3     XPE3           XPEDBMP2    BMP    XPEDPSB2                 IMSA
ASJUSR4     XPE4           XPEDTRA4     TP    XPEDTRA4     004   016   IMSA
ASJUSR5     XPE5           XPEDTRA5     TP    XPEDTRA5     004   018   IMSA
ASJUSR6     XFP1            XPEDFP1    IFP    FASTPAT1     001         IMSA
ASJUSR7     XFP2            XPEDFP2    IFP    FASTPAT2     001         IMSA
ASJUSR8                    XPEDBMP3    BMP    XPEDBMP                  IMSA
ASJUSR9     XPE6           XPEDTRA6     TP    XPEDTRA6     004   019   IMSA
ASJUSR9     XPE7           XPEDTRA7     TP    XPEDTRA7     005   019   IMSA
ASJUSR9     XPE8           XPEDTRA8     TP    XPEDTRA8     005   019   IMSA
ASJUSR10                   XPEDBMP4    BMP    XPEDPSB2                 IMSA

 F1=HELP     F2=SPLIT    F3=END      F4=RETURN   F5=RFIND    F6=RCHANGE
 F7=UP       F8=DOWN     F9=SWAP    F10=LEFT    F11=RIGHT   F12=RETRIEVE
```

Figure B-21. Intercepts Screen

The Intercepts screen does not contain any input fields; all of the information is displayed for you. The fields on the screen are described below:

Number Of Available Class Codes
The number of class codes available. This field tells you how many more people can run an MPP test. Notice that Figure B-21 indicates that the maximum number of MPP users are testing with XPEDITER/IMS by displaying the number of class codes available for use as zero (0).

Current IMSID
The name of the IMS control region with which the current user communicates.

USERID
The user ID of the person running the test.

TRAN CODE
The transaction code associated with the program, if one was used.

PROGRAM
The name of the program being tested.

TYPE
The type of program being tested.

PSB
The program specification block (PSB) name associated with the program. Often, this name is the same as the program name.

OLD CLASS
The original class code of the user's transaction.

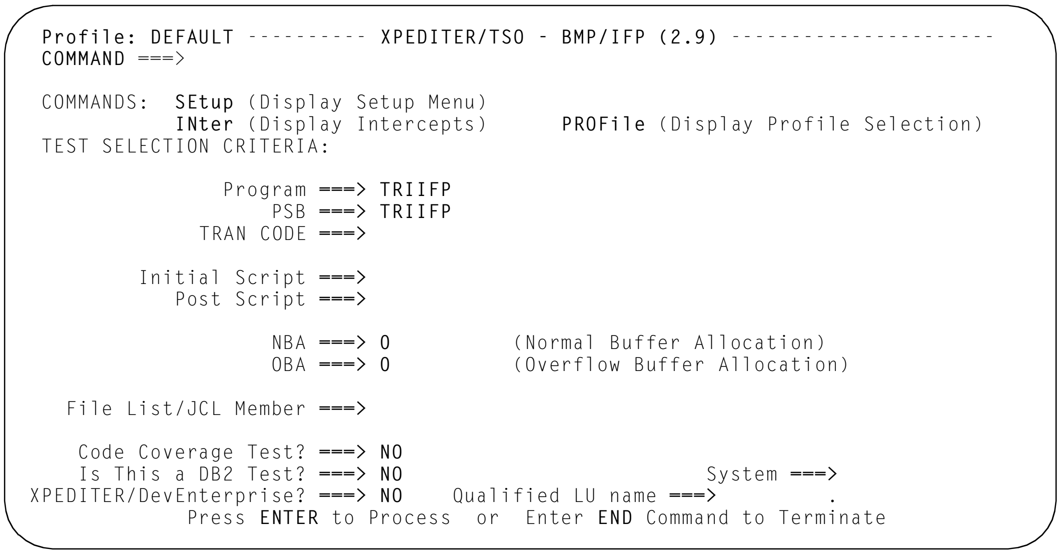
NEW CLASS
The XPEDITER/IMS reserved class code for the transaction. XPEDITER/IMS reassigns the class codes for message processing transactions.

IMSID

The name of the IMS control region with which the IMS user programs communicate. This name does not have to be the current IMSID.

BMP/IFP Test Screen – Environments Menu - Option 9

The BMP/IFP screen displayed in Figure B-22 lets you set up debug parameters before beginning program execution of a program in an IMS, BMP, or Fast Path Region. When you identify the transactions to be debugged and initiate a session, XPEDITER/IMS attaches the IMS message region within the TSO address space. You can enter the transaction code from an IMS terminal to start the transaction, and the source will be displayed on the TSO terminal where the debugging session was initiated. The operation of the product requires one logical TSO terminal and one logical IMS terminal (can be an ATM terminal), both on the same CPU.



```
 Profile: DEFAULT ---------- XPEDITER/TSO - BMP/IFP (2.9) ----------------------
 COMMAND ===>

 COMMANDS:  SEtup (Display Setup Menu)
            INter (Display Intercepts)      PROFile (Display Profile Selection)
 TEST SELECTION CRITERIA:

               Program ===> TRIIFP
                   PSB ===> TRIIFP
             TRAN CODE ===>

        Initial Script ===>
           Post Script ===>

                   NBA ===> 0          (Normal Buffer Allocation)
                   OBA ===> 0          (Overflow Buffer Allocation)

   File List/JCL Member ===>

     Code Coverage Test? ===> NO
     Is This a DB2 Test? ===> NO                           System ===>
XPEDITER/DevEnterprise? ===> NO    Qualified LU name ===>         .
             Press ENTER to Process  or  Enter END Command to Terminate
```

Figure B-22. BMP/IFP Screen

The fields on the BMP/IFP screen are:

Program

Required. Used to identify which program is to be intercepted and debugged by XPEDITER/IMS. You must enter the load module name in this field. XPEDITER inserts this value into the program subparameter for the parameter it passes to IMS.

PSB

Required. Enter the program specification block (PSB) associated with the program you are debugging. XPEDITER inserts the specified value into the BMP parameter passed to the IMS driver program.

TRAN CODE

Enter the optional IMS transaction code. The value specified is inserted into the INPUT TRAN subparameter of the BMP parameter passed to the IMS driver program.

Initial Script

Enter the member name of a test script if you want to execute a predefined command stream at the *beginning* of the debugging session. This member will be executed after the inclusion of the Site-wide script member @@SITE@@, if defined.

Post Script

Enter the member name of a test script if you want to execute a predefined command stream at the *end* of the debugging session.

NBA
Enter the Normal Buffer Allocation (NBA), up to two digits. The default value is 0.

OBA
Enter the Overflow Buffer Allocation (OBA), up to two digits. The default value is 0.

Note: The NBA and OBA fields appear on the screen only if your site installer indicated that your site uses Fast Path databases. The values are inserted in the PARM string when the IMS region is attached.

File List/JCL Member
Enter the dataset name that contains the file list, CLIST, or JCL. The File Allocation Utility (FAU) preallocates files and databases that will be processed by the program upon entry to the debugging session.

If the member name of a PDS is omitted, a member list is displayed.

If the dataset contains a CLIST, XPEDITER/IMS immediately executes it and begins the debugging session.

If the dataset contains a file list or JCL, the FAU is invoked to dynamically allocate the specified files before beginning the debugging session. Refer to Appendix A, "Using the File Allocation Utility" on page A-1 for detailed information about the FAU.

Code Coverage Test?
Enter **YES** if XPEDITER/Code Coverage data should be collected for this test. The default value is NO.

Note: The Code Coverage option is only available if you have purchased XPEDITER/Code Coverage.

Is This a DB2 Test?
Enter **YES** if the program executes EXEC SQL statements. The default value is NO.

System
Enter the DB2 subsystem name if the program executes EXEC SQL statements. The subsystem name depends on the release level of DB2 and is assigned at the time of installation.

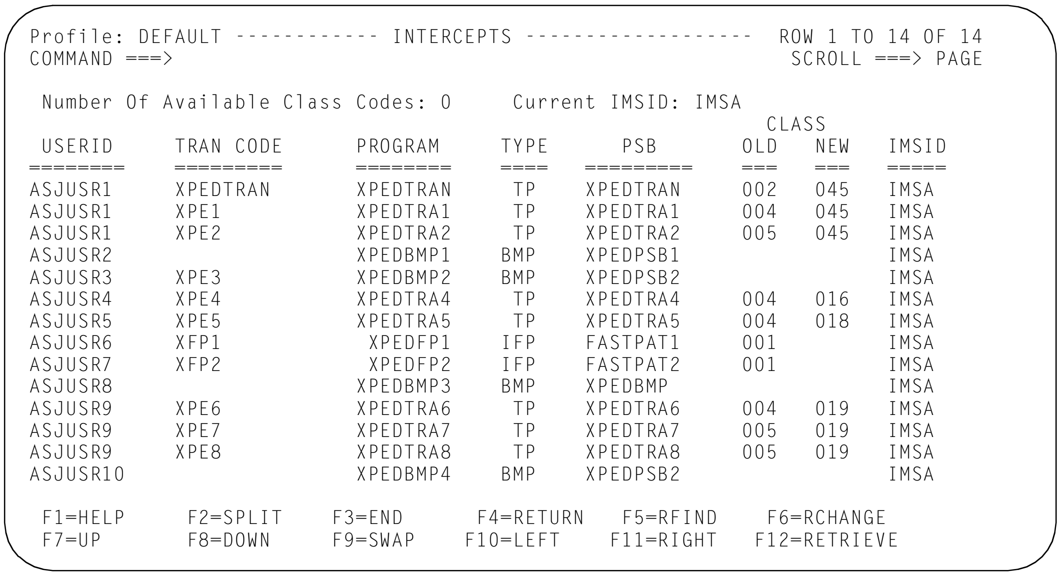
XPEDITER/DevEnterprise?
Enter **YES** in this field if you want to connect to XPEDITER/DevEnterprise. The default is NO.

Qualified LU name
Enter the fully qualified LU name if you are going to use XPEDITER/DevEnterprise. Enter the same name that was entered on the installation screen.

Press **Enter** to initiate the debugging session. Refer to Chapter 5, "Debugging Interactively" on page 5-1 for information on how to debug your program.

When intercepts cannot be set and the message `MAX USERS` is displayed on the test screen, enter INTER on the command line to look at the Intercepts screen, shown in Figure B-23 on page B-29. The Number of Available Class Codes field will be zero (0), indicating that the maximum number of users are testing with XPEDITER/IMS.

The Intercepts screen provides valuable information regarding the use of XPEDITER/IMS—how many users can still test using XPEDITER/IMS, what programs or transactions are being tested, and other information connected with the program and the transaction code.



```
Profile: DEFAULT ----------- INTERCEPTS ------------------  ROW 1 TO 14 OF 14
COMMAND ===>                                                SCROLL ===> PAGE

 Number Of Available Class Codes: 0     Current IMSID: IMSA
                                                              CLASS
 USERID     TRAN CODE      PROGRAM     TYPE      PSB       OLD   NEW   IMSID
========    =========      ========    ====    =========   ===   ===   =====
ASJUSR1     XPEDTRAN       XPEDTRAN     TP     XPEDTRAN    002   045   IMSA
ASJUSR1     XPE1           XPEDTRA1     TP     XPEDTRA1    004   045   IMSA
ASJUSR1     XPE2           XPEDTRA2     TP     XPEDTRA2    005   045   IMSA
ASJUSR2                    XPEDBMP1    BMP     XPEDPSB1                IMSA
ASJUSR3     XPE3           XPEDBMP2    BMP     XPEDPSB2                IMSA
ASJUSR4     XPE4           XPEDTRA4     TP     XPEDTRA4    004   016   IMSA
ASJUSR5     XPE5           XPEDTRA5     TP     XPEDTRA5    004   018   IMSA
ASJUSR6     XFP1            XPEDFP1    IFP     FASTPAT1    001         IMSA
ASJUSR7     XFP2            XPEDFP2    IFP     FASTPAT2    001         IMSA
ASJUSR8                    XPEDBMP3    BMP     XPEDBMP                 IMSA
ASJUSR9     XPE6           XPEDTRA6     TP     XPEDTRA6    004   019   IMSA
ASJUSR9     XPE7           XPEDTRA7     TP     XPEDTRA7    005   019   IMSA
ASJUSR9     XPE8           XPEDTRA8     TP     XPEDTRA8    005   019   IMSA
ASJUSR10                   XPEDBMP4    BMP     XPEDPSB2                IMSA

 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure B-23. Intercepts Screen

The Intercepts screen does not contain any input fields; all of the information is displayed for you. The fields on the screen are described below:

Number Of Available Class Codes
The number of class codes available. This field tells you how many more people can run an MPP test. Notice that Figure B-23 displays the number of class codes available for use as zero (0), indicating that the maximum number of MPP users are testing with XPEDITER/IMS.

Current IMSID
The name of the IMS control region with which the current user communicates.

USERID
The user ID of the person running the test.

TRAN CODE
The transaction code associated with the program, if one was used.

PROGRAM
The name of the program being tested.

TYPE
The type of program being tested.

PSB
The program specification block (PSB) name associated with the program. Often, this name is the same as the program name.

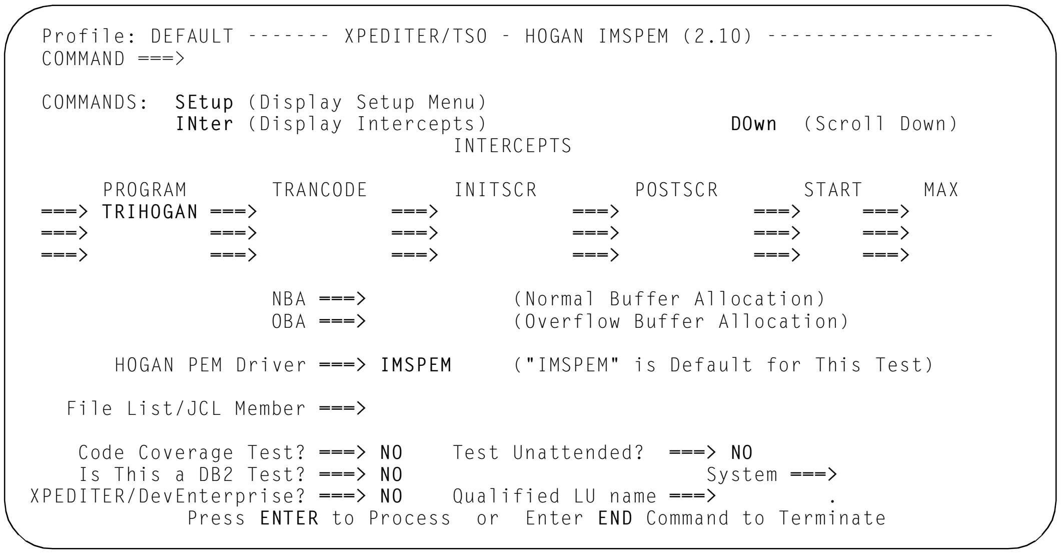
OLD CLASS
The original class code of the user's transaction.

NEW CLASS
The XPEDITER/IMS reserved class code for the transaction. XPEDITER/IMS reassigns the class codes for message processing transactions.

IMSID
The name of the IMS control region with which the IMS user programs communicate. This name does not have to be the current IMSID.

Hogan IMSPEM Test Screen – Environments Menu - Option 10

The Hogan IMSPEM screen displayed in Figure B-24 lets you set up debug parameters before beginning program execution of Hogan IMSPEM in an IMS message region. When you identify the transactions to be debugged and initiate a session, XPEDITER/IMS attaches the IMS message region within the TSO address space. You can enter the transaction code from an IMS terminal to start the transaction, and the source will be displayed on the TSO terminal where the debugging session was initiated. The operation of the product requires one logical TSO terminal and one logical IMS terminal (can be an ATM terminal), both on the same CPU.



```
  Profile: DEFAULT ------- XPEDITER/TSO - HOGAN IMSPEM (2.10) ------------------
  COMMAND ===>

  COMMANDS:  SEtup (Display Setup Menu)
             INter (Display Intercepts)                   DOwn  (Scroll Down)
                                   INTERCEPTS

       PROGRAM       TRANCODE       INITSCR        POSTSCR       START     MAX
  ===> TRIHOGAN ===>          ===>           ===>           ===>      ===>
  ===>          ===>          ===>           ===>           ===>      ===>
  ===>          ===>          ===>           ===>           ===>      ===>

                   NBA ===>            (Normal Buffer Allocation)
                   OBA ===>            (Overflow Buffer Allocation)

        HOGAN PEM Driver ===> IMSPEM     ("IMSPEM" is Default for This Test)

    File List/JCL Member ===>

     Code Coverage Test? ===> NO    Test Unattended?  ===> NO
     Is This a DB2 Test? ===> NO                         System ===>
 XPEDITER/DevEnterprise? ===> NO    Qualified LU name ===>        .
              Press ENTER to Process  or  Enter END Command to Terminate
```

Figure B-24. Hogan IMSPEM Screen

The fields on the Hogan IMSPEM screen are:

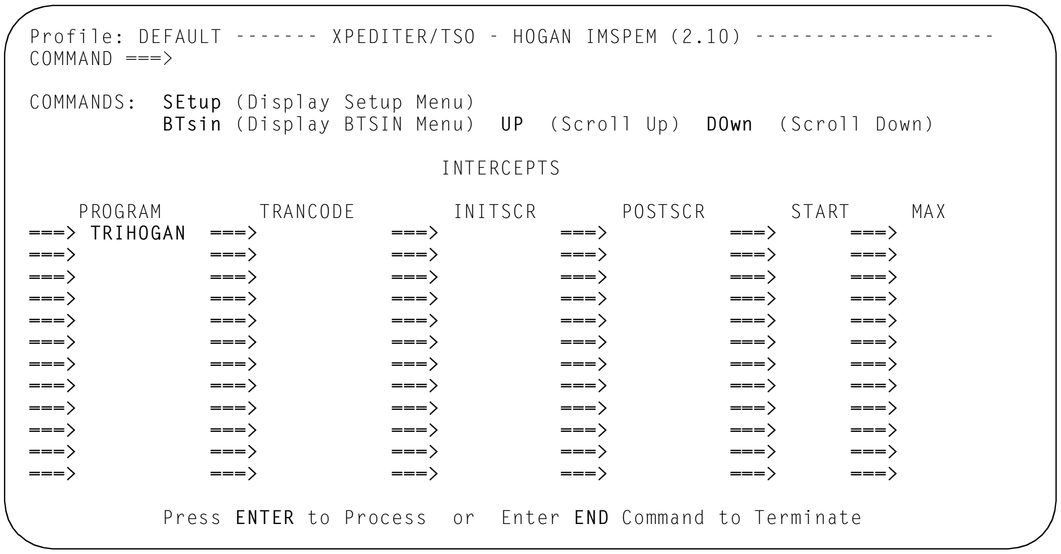
PROGRAM

Required if TRANCODE is not specified. This field is used to identify which programs are to be intercepted and debugged by XPEDITER/IMS. You can enter the load module name in this field, the transaction code in the TRANCODE field, or supply values for both fields. If you enter only the program name (as shown in Figure B-24), all the transactions associated with that program are intercepted and debugged by XPEDITER/IMS. If you enter only a transaction code, its associated program is intercepted.

Three lines of intercept-related data can be specified on this screen. If you want to continue the list, you can scroll down to a second screen, which contains only intercept information.

The second test screen, shown in Figure B-25 on page B-31, must be filled up before the third screen can be called. Note that on the second and subsequent screens, the UP command is available in addition to the DOWN command. Only eleven intercepts are scrolled at a time, and the last entry on the previous page shows at the top.

Note: XPEDITER/IMS processes every intercept specified on the panel and there is a memory overhead associated with each program specified.



```
Profile: DEFAULT ------- XPEDITER/TSO - HOGAN IMSPEM (2.10) -------------------
COMMAND ===>

COMMANDS:  SEtup (Display Setup Menu)
           BTsin (Display BTSIN Menu)  UP  (Scroll Up)  DOwn  (Scroll Down)

                                 INTERCEPTS

    PROGRAM        TRANCODE        INITSCR       POSTSCR       START     MAX
===> TRIHOGAN  ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>
===>           ===>           ===>          ===>          ===>      ===>

           Press ENTER to Process  or  Enter END Command to Terminate
```

Figure B-25. Hogan IMSPEM Test Screen 2

TRANCODE

Required if PROGRAM is not specified. This field is used to identify which transactions are to be intercepted and debugged by XPEDITER/IMS. You can enter the transaction code in this field, the load module name in the PROGRAM field, or supply values for both fields.

Each transaction code must be unique. If you enter only a transaction code, its associated program is intercepted. However, multiple transactions can invoke the same program. Therefore, if you enter only the program name, all the transactions associated with that program are intercepted and debugged by XPEDITER/IMS.

INITSCR

Enter the member name of a test script if you want to execute a predefined command stream at the *beginning* of the debugging session. You can optionally enter a member name from a Setup INCLUDE library. This member will be executed after the inclusion of the Site-wide script member @@SITE@@, if defined.

POSTSCR

Enter the member name of a test script if you want to execute a predefined command stream at the *end* of the debugging session. You can optionally enter a member name from a Setup INCLUDE library.

START

Enter up to four digits; the default value is 1. The start value specifies the intercept on which processing begins. The program's source code is displayed when the start intercept is processed.

MAX

Enter the number of times the intercept is to remain active. If you do not supply a MAX value, XPEDITER/IMS intercepts the transaction (or program) each time it is invoked. If you enter an EXIT command and there are still intercepts remaining, those intercepts are ignored.

NBA

Enter the Normal Buffer Allocation (NBA), up to two digits.

OBA

Enter the Overflow Buffer Allocation (OBA), up to two digits.

Note: The NBA and OBA fields appear on the screen only if your site installer indicated that your site uses Fast Path databases. The values are inserted in the PARM string when the IMS region is attached.

Hogan PEM Driver
Enter the name of the Hogan driver. IMSPEM is the default.

File List/JCL Member
Enter the dataset name that contains the file list, CLIST, or JCL. The File Allocation Utility (FAU) preallocates files and databases that will be processed by the program upon entry to the debugging session.

If the member name of a PDS is omitted, a member list is displayed.

If the dataset contains a CLIST, XPEDITER/IMS immediately executes it and begins the debugging session.

If the dataset contains a file list or JCL, the FAU is invoked to dynamically allocate the specified files before beginning the debugging session. Refer to Appendix A, "Using the File Allocation Utility" on page A-1 for detailed information about the FAU.

Code Coverage Test?
Enter **YES** if XPEDITER/Code Coverage data should be collected for this test. The default value is NO.

Note: The Code Coverage option is only available if you have purchased XPEDITER/Code Coverage.

Test Unattended?
Enter **YES** in the 'Test Unattended?' field to run in Unattended mode. The default value is NO. In unattended mode, after the XPEDITER/IMS Message Processing Region (MPR) is attached, only XPEDITER commands are processed in the Initial Script, Post Script, and Abend Script. The TSO terminal remains locked and XPEDITER commands are not allowed from the terminal. When you are finished testing in the XPEDITER/IMS MPR, the MPR may be stopped and the TSO terminal unlocked by using either the XPEDITER Stop Region transaction, XPST, or the XPEDITER Stop Region BMP procedure XPSTOP. Refer to Chapter 7, "Stopping the XPEDITER/IMS Dependent Region" on page 7-12.

Is This a DB2 Test?
Enter **YES** if the program executes EXEC SQL statements. The default value is NO.

System
Enter the DB2 subsystem name if the program executes EXEC SQL statements. The subsystem name depends on the release level of DB2 and is assigned at the time of installation.

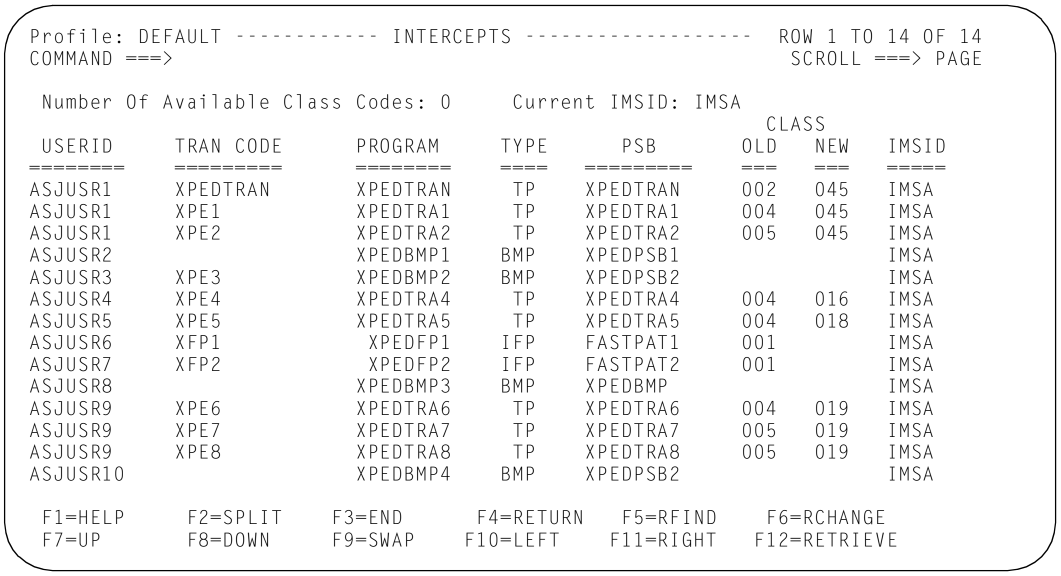
XPEDITER/DevEnterprise?
Enter **YES** in this field if you want to connect to XPEDITER/DevEnterprise. The default is NO.

Qualified LU name
Enter the fully qualified LU name if you are going to use XPEDITER/DevEnterprise. Enter the same name that was entered on the installation screen.

Press **Enter** to initiate the debugging session. Refer to Chapter 5, "Debugging Interactively" on page 5-1 for information on how to debug your program.

When intercepts cannot be set and the message `MAX USERS` is displayed on the test screen, enter INTER on the command line to look at the Intercepts screen, shown in Figure B-26 on page B-33. The Number of Available Class Codes field will be zero (0), indicating that the maximum number of users are testing with XPEDITER/IMS.

The Intercepts screen provides valuable information regarding the use of XPEDITER/IMS—how many users can still test using XPEDITER/IMS, what programs or transactions are being tested, and other information connected with the program and the transaction code.



```
Profile: DEFAULT ----------- INTERCEPTS ------------------  ROW 1 TO 14 OF 14
COMMAND ===>                                                 SCROLL ===> PAGE

 Number Of Available Class Codes: 0     Current IMSID: IMSA
                                                            CLASS
 USERID     TRAN CODE      PROGRAM     TYPE      PSB       OLD   NEW   IMSID
========    =========      ========    ====    =========   ===   ===   =====
ASJUSR1     XPEDTRAN       XPEDTRAN     TP     XPEDTRAN    002   045   IMSA
ASJUSR1     XPE1           XPEDTRA1     TP     XPEDTRA1    004   045   IMSA
ASJUSR1     XPE2           XPEDTRA2     TP     XPEDTRA2    005   045   IMSA
ASJUSR2                    XPEDBMP1    BMP     XPEDPSB1                IMSA
ASJUSR3     XPE3           XPEDBMP2    BMP     XPEDPSB2                IMSA
ASJUSR4     XPE4           XPEDTRA4     TP     XPEDTRA4    004   016   IMSA
ASJUSR5     XPE5           XPEDTRA5     TP     XPEDTRA5    004   018   IMSA
ASJUSR6     XFP1            XPEDFP1    IFP     FASTPAT1    001         IMSA
ASJUSR7     XFP2            XPEDFP2    IFP     FASTPAT2    001         IMSA
ASJUSR8                    XPEDBMP3    BMP     XPEDBMP                 IMSA
ASJUSR9     XPE6           XPEDTRA6     TP     XPEDTRA6    004   019   IMSA
ASJUSR9     XPE7           XPEDTRA7     TP     XPEDTRA7    005   019   IMSA
ASJUSR9     XPE8           XPEDTRA8     TP     XPEDTRA8    005   019   IMSA
ASJUSR10                   XPEDBMP4    BMP     XPEDPSB2                IMSA

 F1=HELP     F2=SPLIT    F3=END      F4=RETURN   F5=RFIND    F6=RCHANGE
 F7=UP       F8=DOWN     F9=SWAP    F10=LEFT    F11=RIGHT   F12=RETRIEVE
```

Figure B-26. Intercepts Screen

The Intercepts screen does not contain any input fields; all of the information is displayed for you. The fields on the screen are described below:

Number Of Available Class Codes
: The number of class codes available. This field tells you how many more people can run an MPP test. Notice that Figure B-26 displays the number of class codes available for use as zero (0), indicating that the maximum number of MPP users are testing with XPEDITER/IMS.

Current IMSID
: The name of the IMS control region with which the current user communicates.

USERID
: The user ID of the person running the test.

TRAN CODE
: The transaction code associated with the program, if one was used.

PROGRAM
: The name of the program being tested.

TYPE
: The type of program being tested.

PSB
: The program specification block (PSB) name associated with the program. Often, this name is the same as the program name.

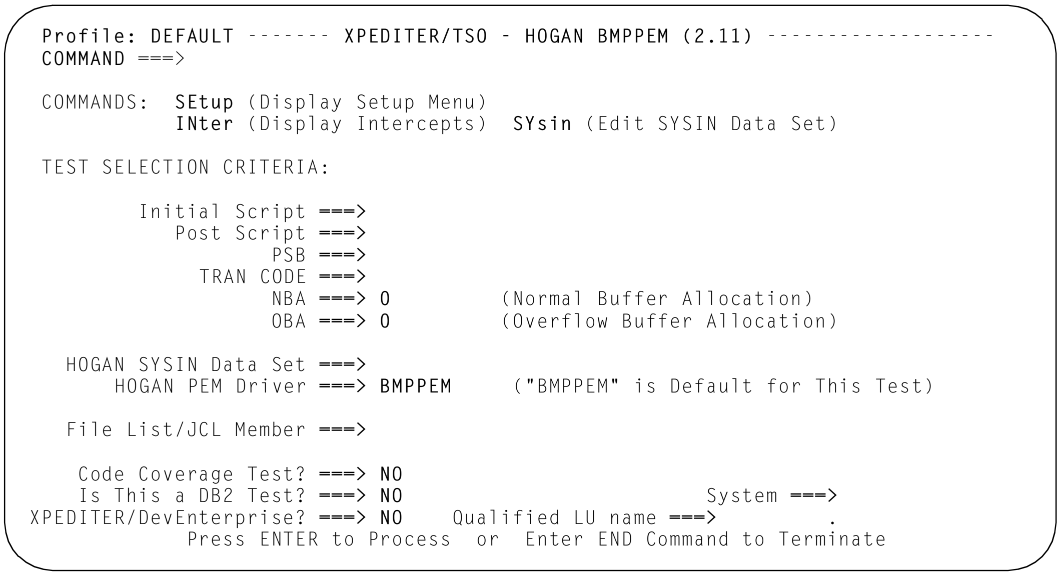
OLD CLASS
: The original class code of the user's transaction.

NEW CLASS
: The XPEDITER/IMS reserved class code for the transaction. XPEDITER/IMS reassigns the class codes for message processing transactions.

IMSID
: The name of the IMS control region with which the IMS user programs communicate. This name does not have to be the current IMSID.

Hogan BMPPEM Test Screen – Environments Menu - Option 11

The Hogan BMPPEM screen displayed in Figure B-27 lets you set up debug parameters before beginning program execution of Hogan BMPPEM in a BMP Region.



```
  Profile: DEFAULT ------- XPEDITER/TSO - HOGAN BMPPEM (2.11) ------------------
  COMMAND ===>

  COMMANDS:  SEtup (Display Setup Menu)
             INter (Display Intercepts)  SYsin (Edit SYSIN Data Set)

  TEST SELECTION CRITERIA:

          Initial Script ===>
             Post Script ===>
                     PSB ===>
               TRAN CODE ===>
                     NBA ===> 0         (Normal Buffer Allocation)
                     OBA ===> 0         (Overflow Buffer Allocation)

    HOGAN SYSIN Data Set ===>
        HOGAN PEM Driver ===> BMPPEM     ("BMPPEM" is Default for This Test)

    File List/JCL Member ===>

     Code Coverage Test? ===> NO
     Is This a DB2 Test? ===> NO                        System ===>
 XPEDITER/DevEnterprise? ===> NO    Qualified LU name ===>         .
              Press ENTER to Process  or  Enter END Command to Terminate
```

Figure B-27. Hogan BMPPEM Screen

The fields on the Hogan BMPPEM screen are:

Initial Script
Enter the member name of a test script if you want to execute a predefined command stream at the *beginning* of the debugging session. You can optionally enter a member name from a Setup INCLUDE library. This member will be executed after the inclusion of the Site-wide script member @@SITE@@, if defined.

Post Script
Enter the member name of a test script if you wish to execute a predefined command stream at the *end* of the debugging session. You can optionally enter a member name from a Setup INCLUDE library.

PSB
Enter the program specification block (PSB) name associated with the program you are debugging.

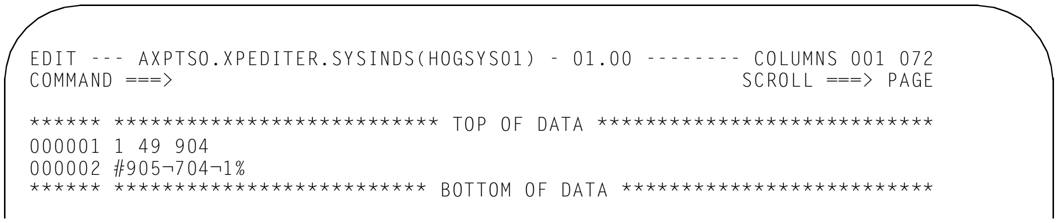
TRAN CODE
Enter the optional IMS input transaction code. This value is inserted into the IMS PARM specified under Setup.

NBA
Enter the Normal Buffer Allocation (NBA), up to two digits. The default value is 0.

OBA
Enter the Overflow Buffer Allocation (OBA), up to two digits. The default value is 0.

Note: The NBA and OBA fields appear on the screen only if your site installer indicated that your site uses Fast Path databases. The values are inserted in the PARM string when the IMS region is attached.

Hogan SYSIN Data Set
A control file used by Hogan programs. Enter **SYSIN** on the command line to edit or create the SYSIN dataset, shown in Figure B-28 on page B-35.



```
EDIT --- AXPTSO.XPEDITER.SYSINDS(HOGSYS01) - 01.00 -------- COLUMNS 001 072
COMMAND ===>                                                 SCROLL ===> PAGE

****** *************************** TOP OF DATA ****************************
000001 1 49 904
000002 #905¬704¬1%
****** ************************** BOTTOM OF DATA **************************
```

Figure B-28. Displaying the SYSIN Dataset

Hogan PEM Driver
Enter the name of the Hogan driver. BMPPEM is the default.

File List/JCL Member
Enter the dataset name that contains the file list, CLIST, or JCL. The File Allocation Utility (FAU) preallocates files and databases that are processed by the program upon entry to the debugging session. Any files or databases that are not dynamically allocated by Hogan must be allocated through this file.

If the member name of a PDS is omitted, a member list is displayed.

If the dataset contains a CLIST, XPEDITER/TSO immediately executes it and begins the debugging session.

If the dataset contains a file list or JCL, the FAU is invoked to dynamically allocate the specified files before beginning the debugging session. Refer to Appendix A, "Using the File Allocation Utility" on page A-1 for detailed information about the FAU.

Note: All CDMF files and databases should be allocated.

Code Coverage Test?
Enter **YES** if XPEDITER/Code Coverage data should be collected for this test. The default value is NO.

Note: The Code Coverage option is only available if you have purchased XPEDITER/Code Coverage.

Is This a DB2 Test?
Enter **YES** if the program executes EXEC SQL statements. The default value is NO.

System
Enter the DB2 subsystem name if the program executes EXEC SQL statements. The subsystem name depends on the release level of DB2 and is assigned at the time of installation.

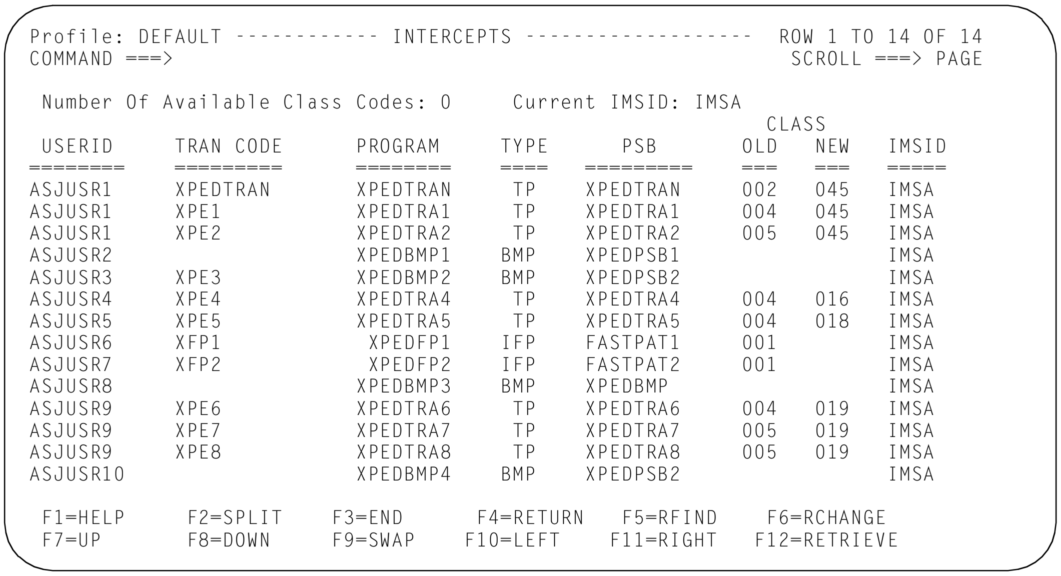
XPEDITER/DevEnterprise?
Enter **YES** in this field if you want to connect to XPEDITER/DevEnterprise. The default is NO.

Qualified LU name
Enter the fully qualified LU name if you are going to use XPEDITER/DevEnterprise. Enter the same name that was entered on the installation screen.

Press **Enter** to initiate the debugging session. Refer to Chapter 5, "Debugging Interactively" on page 5-1 for information on how to debug your program.

When intercepts cannot be set and the message `MAX USERS` is displayed on the test screen, enter INTER on the command line to look at the Intercepts screen, shown in Figure B-29 on page B-36. The Number of Available Class Codes field will be zero (0), indicating that the maximum number of users are testing with XPEDITER/IMS.

The Intercepts screen provides valuable information regarding the use of XPEDITER/IMS—how many users can still test using XPEDITER/IMS, what programs or transactions are being tested, and other information connected with the program and the transaction code.



```
Profile: DEFAULT ----------- INTERCEPTS ------------------  ROW 1 TO 14 OF 14
COMMAND ===>                                                 SCROLL ===> PAGE

 Number Of Available Class Codes: 0     Current IMSID: IMSA
                                                              CLASS
 USERID     TRAN CODE      PROGRAM     TYPE      PSB       OLD   NEW   IMSID
========    =========      ========    ====   =========    ===   ===   =====
ASJUSR1     XPEDTRAN       XPEDTRAN     TP    XPEDTRAN     002   045   IMSA
ASJUSR1     XPE1           XPEDTRA1     TP    XPEDTRA1     004   045   IMSA
ASJUSR1     XPE2           XPEDTRA2     TP    XPEDTRA2     005   045   IMSA
ASJUSR2                    XPEDBMP1    BMP    XPEDPSB1                 IMSA
ASJUSR3     XPE3           XPEDBMP2    BMP    XPEDPSB2                 IMSA
ASJUSR4     XPE4           XPEDTRA4     TP    XPEDTRA4     004   016   IMSA
ASJUSR5     XPE5           XPEDTRA5     TP    XPEDTRA5     004   018   IMSA
ASJUSR6     XFP1            XPEDFP1    IFP    FASTPAT1     001         IMSA
ASJUSR7     XFP2            XPEDFP2    IFP    FASTPAT2     001         IMSA
ASJUSR8                    XPEDBMP3    BMP    XPEDBMP                  IMSA
ASJUSR9     XPE6           XPEDTRA6     TP    XPEDTRA6     004   019   IMSA
ASJUSR9     XPE7           XPEDTRA7     TP    XPEDTRA7     005   019   IMSA
ASJUSR9     XPE8           XPEDTRA8     TP    XPEDTRA8     005   019   IMSA
ASJUSR10                   XPEDBMP4    BMP    XPEDPSB2                 IMSA

 F1=HELP     F2=SPLIT    F3=END      F4=RETURN   F5=RFIND    F6=RCHANGE
 F7=UP       F8=DOWN     F9=SWAP    F10=LEFT    F11=RIGHT   F12=RETRIEVE
```

Figure B-29. Intercepts Screen

The Intercepts screen does not contain any input fields; all of the information is displayed for your analysis. The fields on the screen are described below:

Number Of Available Class Codes
The number of class codes available. This field tells you how many more people can run a MPP test. Notice that Figure B-29 displays the number of class codes available for use as zero (0), indicating that the maximum number of MPP users are testing with XPEDITER/IMS.

Current IMSID
The name of the IMS control region with which the current user communicates.

USERID
The user ID of the person running the test.

TRAN CODE
The transaction code associated with the program, if one was used.

PROGRAM
The name of the program being tested.

TYPE
The type of program being tested.

PSB
The program specification block (PSB) name associated with the program. Often, this name is the same as the program name.

OLD CLASS
The original class code of the user’s transaction.

NEW CLASS
The XPEDITER/IMS reserved class code for the transaction. XPEDITER/IMS reassigns the class codes for message processing transactions.

IMSID
The name of the IMS control region with which the IMS user programs communicate. This name does not have to be the current IMSID.

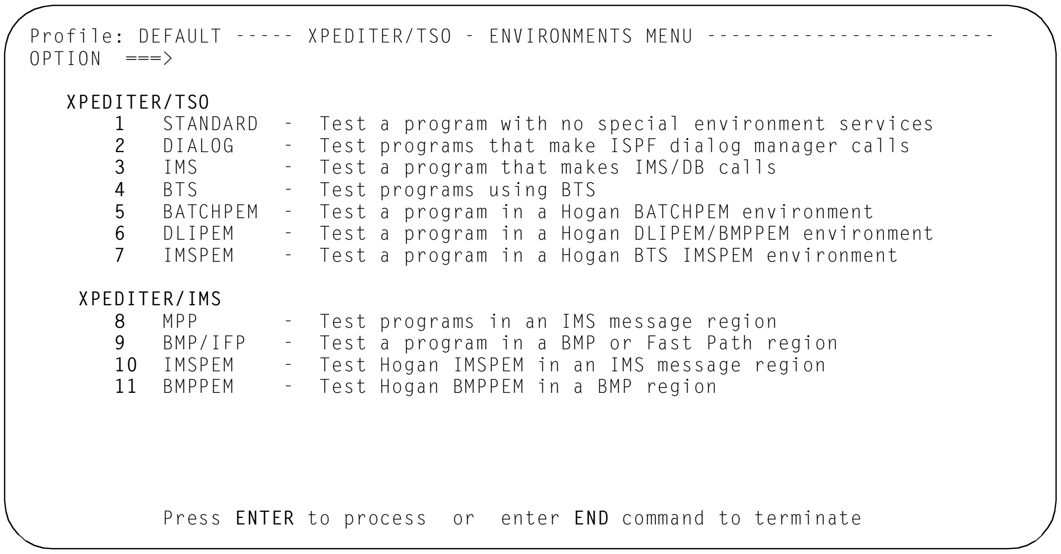
Appendix C. Specifying Setup Options

Setup is a user initialization procedure that lets you assign values to installed options such as: (1) application load library names, (2) DDIO file names, (3) test script library names, and so on. These options automatically default to the installed options the first time you access XPEDITER/TSO; however, you can override the installed options by using the setup procedure. The options you specify are retained across debugging sessions until you change them.

Note: At least one load library and one DDIO library are required for XPEDITER/TSO.

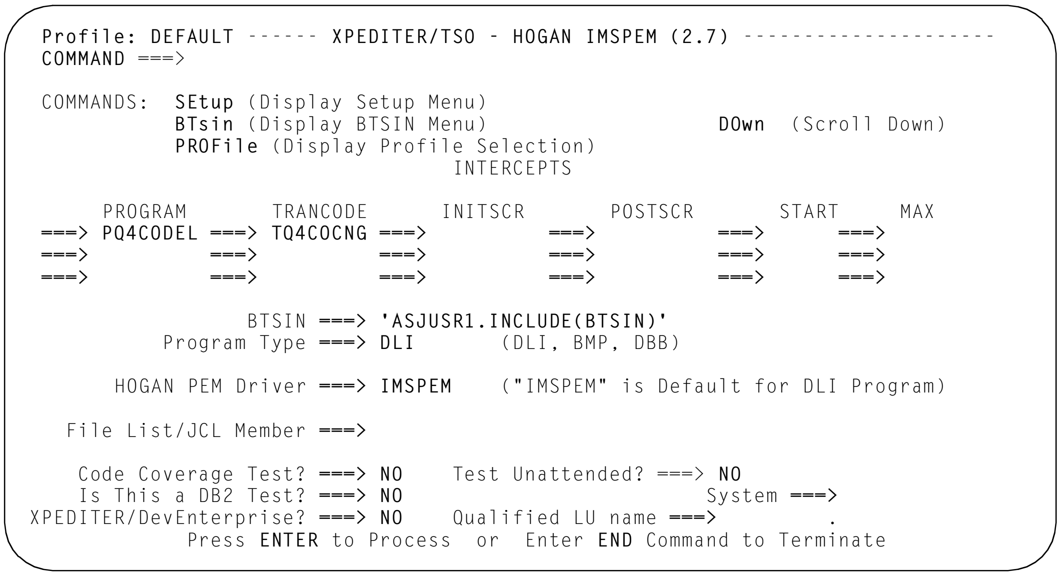
The setup procedure is accessed by typing **SETUP** or **SE** on the command line of any environment test screen or any screen on which the command appears. The options displayed on a Setup Menu depend on the environment in which you run your program and may vary for different environments.

For example, if you selected option **7** (IMSPEM) on the XPEDITER/TSO Environments Menu shown in Figure C-1, the Hogan IMSPEM environment test screen shown in Figure C-2 on page C-2 is displayed.



```
Profile: DEFAULT ----- XPEDITER/TSO - ENVIRONMENTS MENU -----------------------
OPTION  ===>

   XPEDITER/TSO
       1   STANDARD  -  Test a program with no special environment services
       2   DIALOG    -  Test programs that make ISPF dialog manager calls
       3   IMS       -  Test a program that makes IMS/DB calls
       4   BTS       -  Test programs using BTS
       5   BATCHPEM  -  Test a program in a Hogan BATCHPEM environment
       6   DLIPEM    -  Test a program in a Hogan DLIPEM/BMPPEM environment
       7   IMSPEM    -  Test a program in a Hogan BTS IMSPEM environment

    XPEDITER/IMS
       8   MPP       -  Test programs in an IMS message region
       9   BMP/IFP   -  Test a program in a BMP or Fast Path region
      10   IMSPEM    -  Test Hogan IMSPEM in an IMS message region
      11   BMPPEM    -  Test Hogan BMPPEM in a BMP region




           Press ENTER to process  or  enter END command to terminate
```

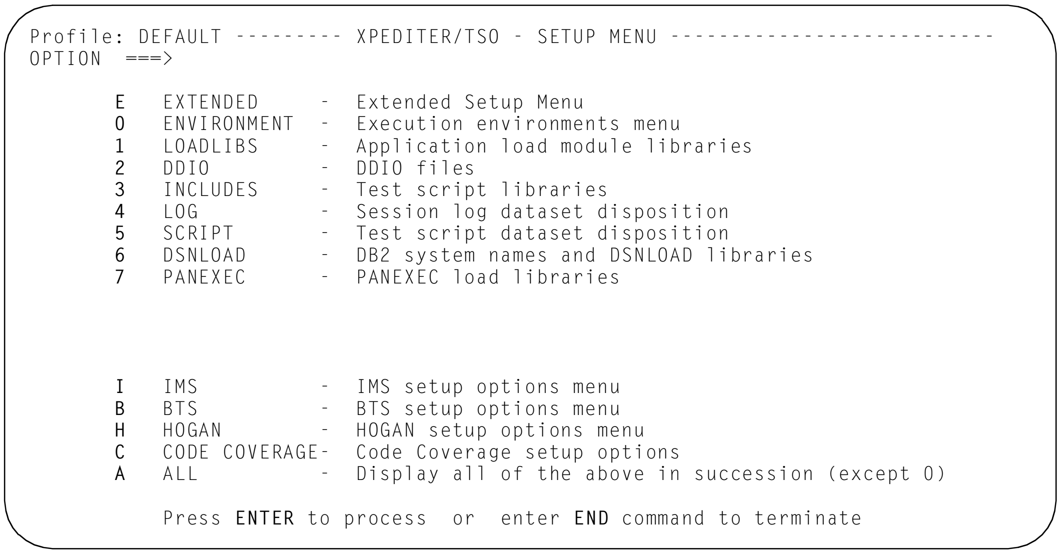
Figure C-1. XPEDITER/TSO Environments Menu



```
 Profile: DEFAULT ------ XPEDITER/TSO - HOGAN IMSPEM (2.7) --------------------
 COMMAND ===>

 COMMANDS:  SEtup (Display Setup Menu)
            BTsin (Display BTSIN Menu)                  DOwn  (Scroll Down)
            PROFile (Display Profile Selection)
                                   INTERCEPTS

      PROGRAM       TRANCODE      INITSCR       POSTSCR       START     MAX
 ===> PQ4CODEL ===> TQ4COCNG ===>          ===>          ===>      ===>
 ===>          ===>          ===>          ===>          ===>      ===>
 ===>          ===>          ===>          ===>          ===>      ===>

                 BTSIN ===> 'ASJUSR1.INCLUDE(BTSIN)'
          Program Type ===> DLI       (DLI, BMP, DBB)

       HOGAN PEM Driver ===> IMSPEM    ("IMSPEM" is Default for DLI Program)

   File List/JCL Member ===>

    Code Coverage Test? ===> NO    Test Unattended? ===> NO
    Is This a DB2 Test? ===> NO                     System ===>
XPEDITER/DevEnterprise? ===> NO    Qualified LU name ===>        .
           Press ENTER to Process  or  Enter END Command to Terminate
```

Figure C-2. Hogan IMSPEM Test Screen

If you enter **SETUP** on the environment test screen, the IMSPEM Setup Menu shown in Figure C-3 is displayed.



```
Profile: DEFAULT --------- XPEDITER/TSO - SETUP MENU --------------------------
OPTION  ===>

       E   EXTENDED     -  Extended Setup Menu
       0   ENVIRONMENT  -  Execution environments menu
       1   LOADLIBS     -  Application load module libraries
       2   DDIO         -  DDIO files
       3   INCLUDES     -  Test script libraries
       4   LOG          -  Session log dataset disposition
       5   SCRIPT       -  Test script dataset disposition
       6   DSNLOAD      -  DB2 system names and DSNLOAD libraries
       7   PANEXEC      -  PANEXEC load libraries




       I   IMS          -  IMS setup options menu
       B   BTS          -  BTS setup options menu
       H   HOGAN        -  HOGAN setup options menu
       C   CODE COVERAGE-  Code Coverage setup options
       A   ALL          -  Display all of the above in succession (except 0)

           Press ENTER to process  or  enter END command to terminate
```

Figure C-3. XPEDITER/TSO IMSPEM Setup Menu

Option 0 on all Setup Menus displays the Environments Menu. Options 1 through 7 on the Setup menus are available for all environments and are referred to as the common setup options. The screens for each of these seven options are described in “Common Setup Screens” on page C-4.

Note: As a convenience, you can bypass the Setup Menus by using the SETUP command and specifying the setup option you want. For example, **SETUP 1** takes you directly to the Load Module Libraries screen, **SE A** automatically cycles you through all the setup screens, and **SETUP 0.4** changes your test type to BTS.

This appendix shows examples of the Setup Menus, and the screens and submenus that can be accessed from them. Since many of the Setup Menus are the same, all of the menus are not shown.

Setup Options Available Under XPEDITER/TSO Environments

Table C-1 shows the setup options that are available for each XPEDITER/TSO environment.

Setup Options	XPEDITER/TSO Environments						
	Standard	Dialog	IMS	BTS	BATCHPEM	DLIPEM	IMSPEM
LOADLIBS	X	X	X	X	X	X	X
DDIO	X	X	X	X	X	X	X
INCLUDES	X	X	X	X	X	X	X
LOG	X	X	X	X	X	X	X
SCRIPT	X	X	X	X	X	X	X
DSNLOAD	X	X	X	X	X	X	X
PANEXEC	X	X	X	X	X	X	X
IMS			X	X		X	X
BTS				X			X
HOGAN					X	X	X
CODE COVERAGE	X	X	X	X	X	X	X

Table C-1. Setup Options Available for XPEDITER/TSO Environments.

Setup Options Available Under XPEDITER/IMS Environments

Table C-2 on page C-4 shows the setup options that are available for each XPEDITER/IMS environment. The Setup Menu shown in Figure C-4 is displayed when you select option **11** (BMPPEM) under XPEDITER/IMS on the Environments Menu and type **SETUP** in the command line of the displayed environment test screen.

Refer to “Common Setup Screens” on page C-4 for a description of the first seven options (options 1 through 7) on the Setup Menu. Refer to "IMS Setup Menu" on page C-16 for descriptions of options 9 through 12 and refer also to "Hogan Setup Menu" on page C-35 for option H.

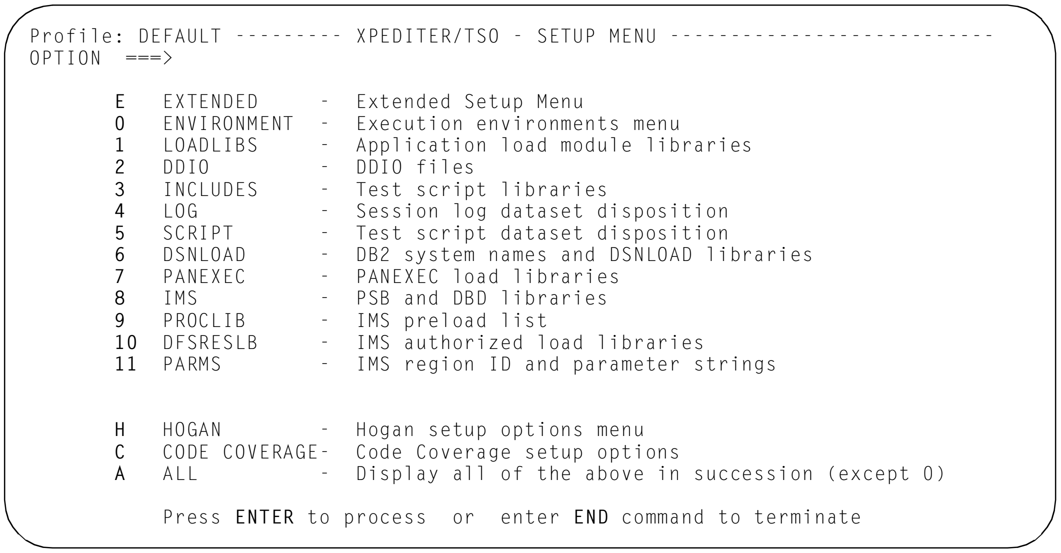

```
Profile: DEFAULT --------- XPEDITER/TSO - SETUP MENU ---------------------------
OPTION  ===>

       E   EXTENDED     -  Extended Setup Menu
       0   ENVIRONMENT  -  Execution environments menu
       1   LOADLIBS     -  Application load module libraries
       2   DDIO         -  DDIO files
       3   INCLUDES     -  Test script libraries
       4   LOG          -  Session log dataset disposition
       5   SCRIPT       -  Test script dataset disposition
       6   DSNLOAD      -  DB2 system names and DSNLOAD libraries
       7   PANEXEC      -  PANEXEC load libraries
       8   IMS          -  PSB and DBD libraries
       9   PROCLIB      -  IMS preload list
       10  DFSRESLB     -  IMS authorized load libraries
       11  PARMS        -  IMS region ID and parameter strings


       H   HOGAN        -  Hogan setup options menu
       C   CODE COVERAGE-  Code Coverage setup options
       A   ALL          -  Display all of the above in succession (except 0)

           Press ENTER to process  or  enter END command to terminate
```

Figure C-4. XPEDITER/IMS BMPPEM Setup Menu

The following notes list the differences between Figure C-4 on page C-3 and the Setup Menu for the different XPEDITER/IMS environments:

Notes:

3. Options 8 (MPP) and 9 (BMP/IFP) on the Environments Menu do not display option H (HOGAN).
4. Option 10 (IMSPEM) on the Environments Menu displays a Setup Menu similar to Figure C-4.

Setup Options	XPEDITER/IMS Environments			
	MPP	BMP/IFP	IMSPEM	BMPPEM
LOADLIBS	X	X	X	X
DDIO	X	X	X	X
INCLUDES	X	X	X	X
LOG	X	X	X	X
SCRIPT	X	X	X	X
DSNLOAD	X	X	X	X
IMS	X	X	X	X
PROCLIB	X	X	X	X
DFSRESLB	X	X	X	X
PARMS	X	X	X	X
HOGAN			X	X
CODE COVERAGE	X	X	X	X

Table C-2. Setup Options Available for XPEDITER/IMS Environments.

Using the RESTORE Command

When you type over an installed default value on a setup screen, that default value is permanently updated with the new value until either a RESTORE command is executed or another update is made to the same field.

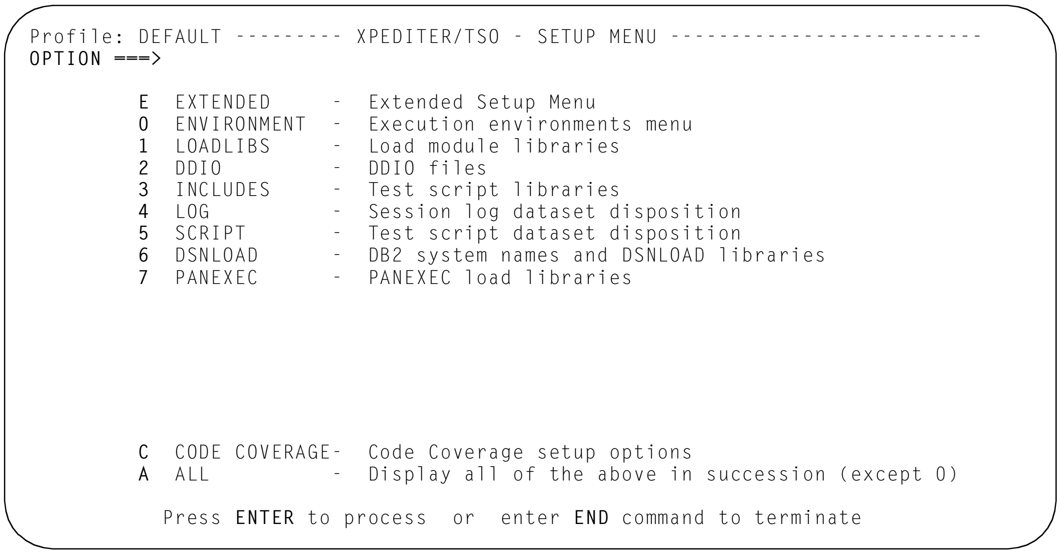
You can use the RESTORE command to reinstate the installed defaults for a *single* screen whose values have been changed or for *all* Setup and installation screens whose values have been changed.

To restore installed defaults for a single setup screen, enter **RESTORE** on the command line of that particular setup screen. RESTORE reinstates all installed default values for the screen and saves them automatically. You cannot undo the RESTORE when you enter it for a single setup screen.

To restore installed defaults *globally* for all setup and installation screens, enter **RESTORE** on the command line of the Setup Menu screen. A confirmation screen is displayed to remind you that the RESTORE command reinstates all installed default values for all setup and installation screens. When the confirmation screen is displayed, press **Enter** to execute the RESTORE command. If you decide to cancel the RESTORE command, enter the **END** command and exit this screen without restoring the installed default values.

Common Setup Screens

If you selected option **1** (STANDARD) on the Environments Menu and entered **SETUP** on the environment test screen, the Standard Setup Menu screen shown in Figure C-5 on page C-5 is displayed.



```
Profile: DEFAULT --------- XPEDITER/TSO - SETUP MENU -------------------------
OPTION ===>

         E  EXTENDED      -  Extended Setup Menu
         0  ENVIRONMENT   -  Execution environments menu
         1  LOADLIBS      -  Load module libraries
         2  DDIO          -  DDIO files
         3  INCLUDES      -  Test script libraries
         4  LOG           -  Session log dataset disposition
         5  SCRIPT        -  Test script dataset disposition
         6  DSNLOAD       -  DB2 system names and DSNLOAD libraries
         7  PANEXEC       -  PANEXEC load libraries








         C  CODE COVERAGE-  Code Coverage setup options
         A  ALL           -  Display all of the above in succession (except 0)

           Press ENTER to process  or  enter END command to terminate
```

Figure C-5. Standard Setup Menu Screen

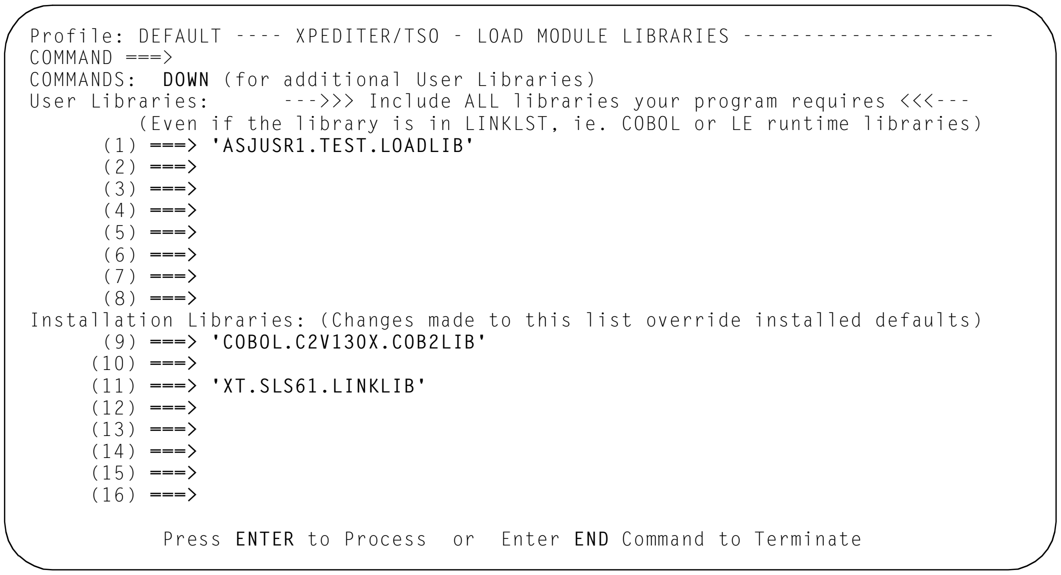
The first six options on the Setup Menu are available for all execution environments. Many of the screens are optional. You must specify at least one load library and one DDIO library. Log and script files are created by XPEDITER/TSO, so you will also need to set up datasets for them. If you don't use scripts, you do not have to specify an INCLUDE library. The screens for each of these options are described in the following subsections.

Load Module Libraries Screens

A load module library list is a set of partitioned datasets containing your link-edited application programs. XPEDITER/TSO searches the list for the modules you want to call or debug during the debugging session. The load module library containing any module you want to intercept (i.e., set a breakpoint on) must be listed on this screen.

The Load Module Libraries screens are used to enter the dsnames of common load libraries needed for most debugging sessions. Up to 32 libraries can be concatenated in this load library list (24 on the User Library list and 8 on the Installation Library list).

The first screen you see is one of three screens. On the first one (as shown in Figure C-6), there is room for eight user libraries and eight installation libraries.

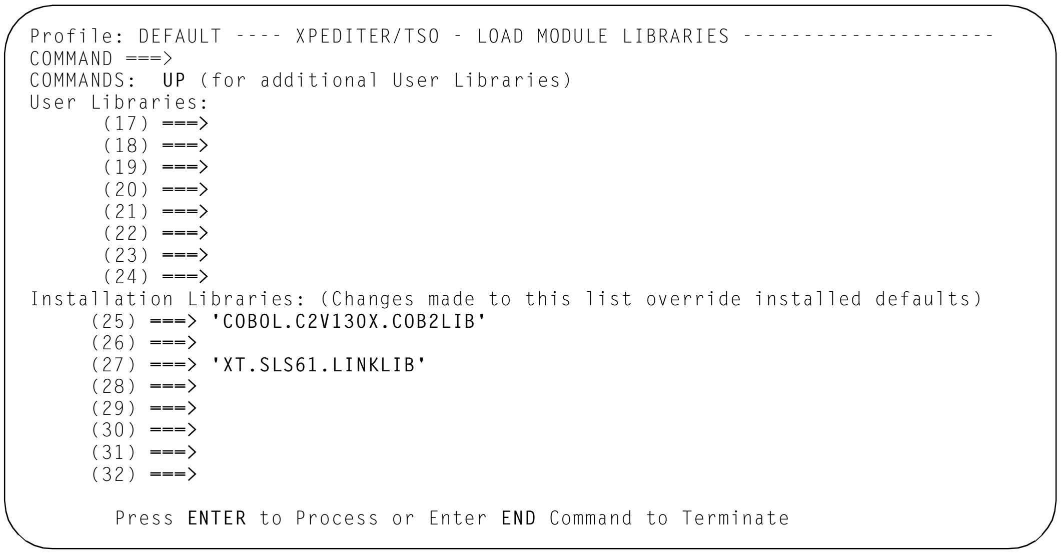


```
Profile: DEFAULT ---- XPEDITER/TSO - LOAD MODULE LIBRARIES ---------------------
COMMAND ===>
COMMANDS:  DOWN (for additional User Libraries)
User Libraries:      --->>> Include ALL libraries your program requires <<<---
         (Even if the library is in LINKLST, ie. COBOL or LE runtime libraries)
      (1) ===> 'ASJUSR1.TEST.LOADLIB'
      (2) ===>
      (3) ===>
      (4) ===>
      (5) ===>
      (6) ===>
      (7) ===>
      (8) ===>
Installation Libraries: (Changes made to this list override installed defaults)
      (9) ===> 'COBOL.C2V130X.COB2LIB'
     (10) ===>
     (11) ===> 'XT.SLS61.LINKLIB'
     (12) ===>
     (13) ===>
     (14) ===>
     (15) ===>
     (16) ===>

           Press ENTER to Process  or  Enter END Command to Terminate
```

Figure C-6. First Load Module Libraries Screen

If you need to list additional user libraries, enter **DOWN** or press the **PF8** key. A second screen appears. This screen lists the user libraries you have already entered and space for eight more libraries. DOWN is still active and you can also enter UP or press the PF7 key to return to the previous screen.

Enter **DOWN** to receive a third screen (Figure C-7), which has space for eight additional user libraries and a list of the original installation libraries. On this third screen, you can enter the UP command to return to the previous screen.



```
Profile: DEFAULT ---- XPEDITER/TSO - LOAD MODULE LIBRARIES --------------------
COMMAND ===>
COMMANDS:  UP (for additional User Libraries)
User Libraries:
      (17) ===>
      (18) ===>
      (19) ===>
      (20) ===>
      (21) ===>
      (22) ===>
      (23) ===>
      (24) ===>
Installation Libraries: (Changes made to this list override installed defaults)
     (25) ===> 'COBOL.C2V130X.COB2LIB'
     (26) ===>
     (27) ===> 'XT.SLS61.LINKLIB'
     (28) ===>
     (29) ===>
     (30) ===>
     (31) ===>
     (32) ===>

       Press ENTER to Process or Enter END Command to Terminate
```

Figure C-7. Third Load Module Libraries Screen

Values for the two fields on these screens are described below.

User Libraries
You can specify up to 24 libraries. Normal concatenation rules are in effect—libraries specified first will be searched first, and buffer space for XPEDITER's task library will be determined by the first library specified.

Installation Libraries
The installer entered the DSNAMEs of common load libraries that should be allocated to any debugging session. Usually, only COBOL subroutines and in-house utility libraries are listed here. You can override installed default libraries by specifying new libraries in these fields.

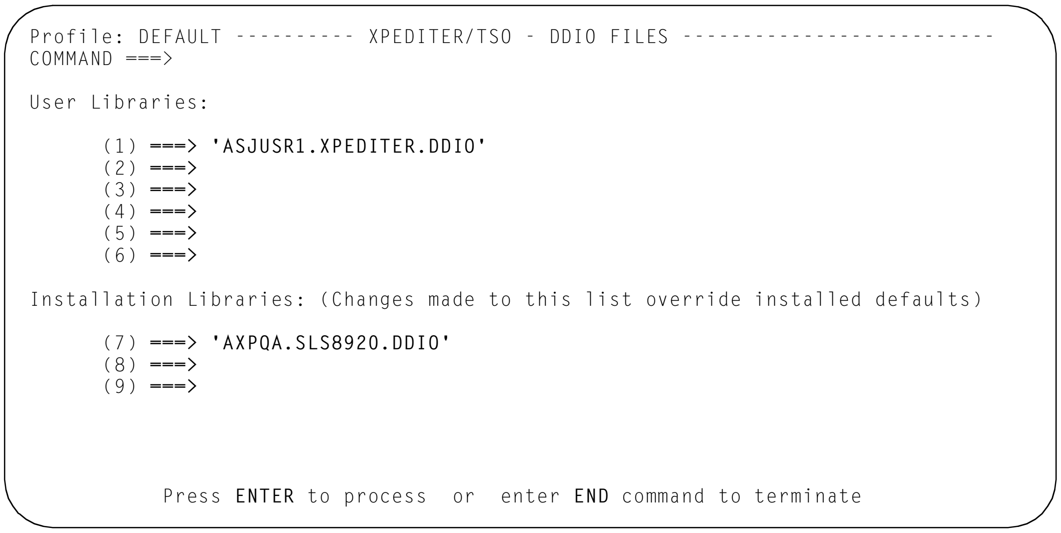
The application load libraries entered in the User Libraries fields are concatenated on top of any libraries entered in these fields.

Usage Note

Even if your application programs would normally find any required Language dependent run-time subroutines (including LE - Language Environment), without being included in the JOBLIB/STEPLIB of the batch JCL (usually from the LINKLIST or (E)LPA), the libraries must still be specified as part of the test session setup. This will ensure that XPEDITER's Task Library will be properly configured.

DDIO Files Screen

The DDIO Files screen is used to enter the names of your DDIO libraries. **At least one DDIO dataset is required.**



```
Profile: DEFAULT --------- XPEDITER/TSO - DDIO FILES -------------------------
COMMAND ===>

User Libraries:

      (1) ===> 'ASJUSR1.XPEDITER.DDIO'
      (2) ===>
      (3) ===>
      (4) ===>
      (5) ===>
      (6) ===>

Installation Libraries: (Changes made to this list override installed defaults)

      (7) ===> 'AXPQA.SLS8920.DDIO'
      (8) ===>
      (9) ===>



           Press ENTER to process  or  enter END command to terminate
```

Figure C-8. DDIO Files Screen

The fields on this screen are:

User Libraries

Some sites find it useful to create multiple DDIO libraries based on user, project, or some other criteria. You can enter the name of your DDIO dataset in these fields.

Fill in the dsname of the VSAM or sequential dataset containing the source listing member for your program (load module). Up to six DDIO libraries can be concatenated.

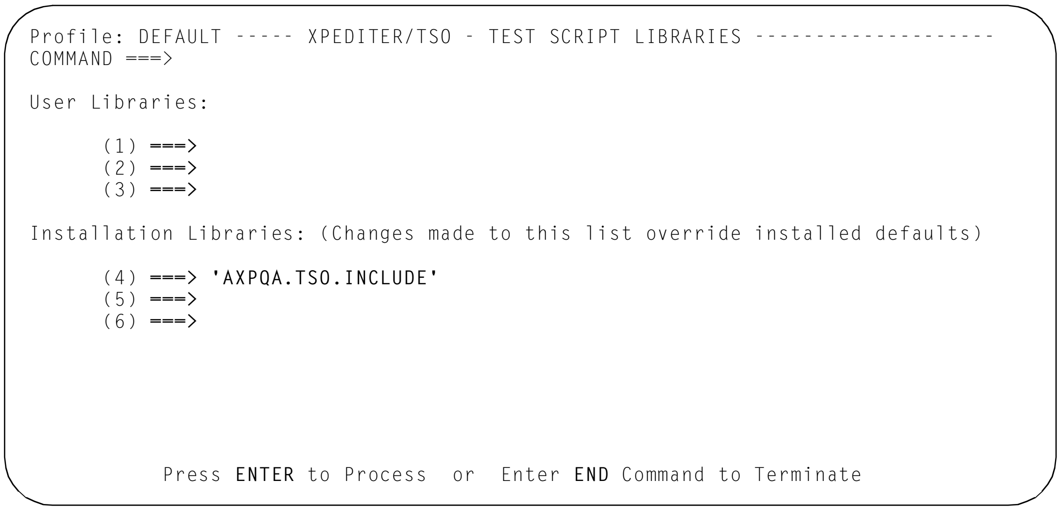
Installation Libraries

If your site has a common DDIO dataset, the installer entered its dsname in this field. At least one library dsname is required before XPEDITER can be invoked, but up to three libraries can be concatenated.

Test Script Libraries Screen

The test script library contains sets of XPEDITER/TSO command streams used to set up, run, or rerun a debugging session. You can copy these command streams into a debugging session with the INCLUDE command. Each script dataset **must** be partitioned, and is allocated to the XINCLUDE ddname.

While you are executing or debugging your program, XPEDITER/TSO automatically generates a script of all the commands entered during the debugging session.



```
Profile: DEFAULT ----- XPEDITER/TSO - TEST SCRIPT LIBRARIES -------------------
COMMAND ===>

User Libraries:

      (1) ===>
      (2) ===>
      (3) ===>

Installation Libraries: (Changes made to this list override installed defaults)

      (4) ===> 'AXPQA.TSO.INCLUDE'
      (5) ===>
      (6) ===>




            Press ENTER to Process  or  Enter END Command to Terminate
```

Figure C-9. Test Script Libraries Screen

You are not required to provide the dsname of a test script library unless you specify an Initial or Post Script on the test screen, or use the INCLUDE command referencing a member not contained in a Site-wide library (specified at installation).

The fields on this screen are:

User Libraries
The user test scripts library is created by you. You can create a member by copying the script dataset created during a debugging session.

Installation Libraries
The installer may have entered the DSN of a test script library included on the installation tape. These sample INCLUDE scripts can be used during installation of XPEDITER/TSO and for subsequent verification and training.

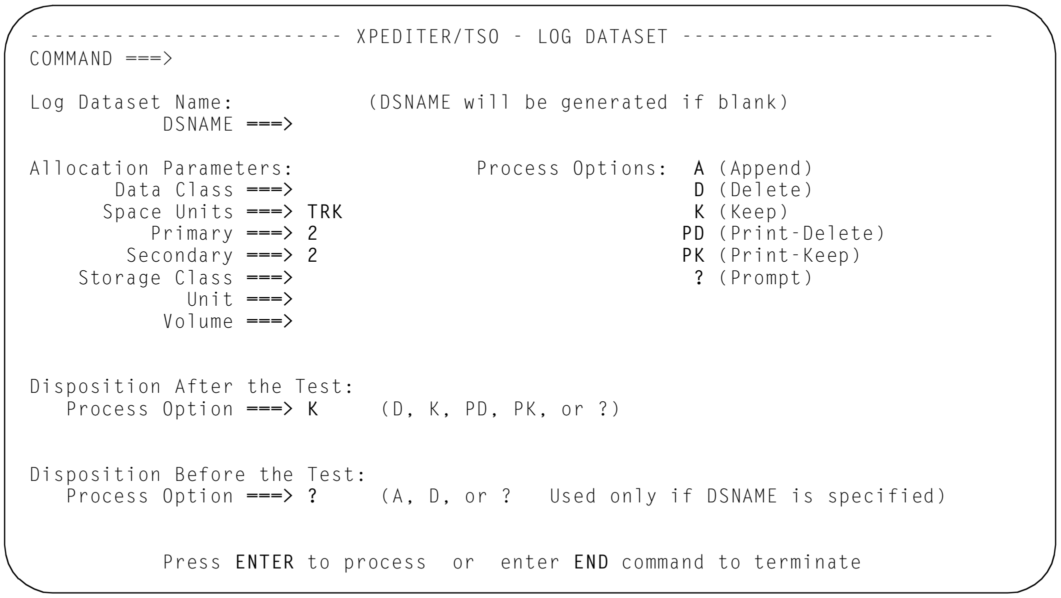
Log and Script Dataset Screens

The Log Dataset and the Script Dataset screens contain the same fields, as do the Log Dataset (2) and Script Dataset (2) screens. Therefore, the fields displayed on each set of screens are discussed only once. When the text in this section and the next refer to log or the log dataset, you can assume it refers also to script, if these four screens are involved.

Although both log and script datasets are automatically created for every debugging session, they contain two different kinds of information at the *end* of the debugging session:

- The *session log* contains a record of the XPEDITER/TSO commands entered during the debugging session and the responses to them.
- The *script dataset* contains each executable command entered in the debugging session. Before the debugging session is terminated, XPEDITER/TSO lets you copy the test script into a test script library. This predefined stream of XPEDITER/TSO commands can then be used to set up, run, or rerun another debugging session.

The Log Dataset and Script Dataset screens let you display and modify the default values for the log and script datasets. Default values are supplied for the dsname and allocation parameters. The disposition of the log and test script *before* and *after* a debugging session must be specified because a log and a test script are created for every session.



```
------------------------- XPEDITER/TSO - LOG DATASET -------------------------
COMMAND ===>

Log Dataset Name:           (DSNAME will be generated if blank)
           DSNAME ===>

Allocation Parameters:               Process Options:  A (Append)
       Data Class ===>                                  D (Delete)
      Space Units ===> TRK                              K (Keep)
          Primary ===> 2                               PD (Print-Delete)
        Secondary ===> 2                               PK (Print-Keep)
    Storage Class ===>                                  ? (Prompt)
             Unit ===>
           Volume ===>


Disposition After the Test:
   Process Option ===> K     (D, K, PD, PK, or ?)


Disposition Before the Test:
   Process Option ===> ?     (A, D, or ?   Used only if DSNAME is specified)


           Press ENTER to process  or  enter END command to terminate
```

Figure C-10. Log/Script Dataset Screen

The screen fields shown in Figure C-10 are described as follows:

Log Dataset Name

If you leave the DSNAME field blank, XPEDITER/TSO generates a log dataset dsname of '*userid*.XPLOG.MONDD.THHMM' (or a script dataset name of '*userid*.XPSCR.MONDD.THHMM'), where MON is a 3-character abbreviation for the current month, DD is the date in the month, and HHMM is the hour and minute the dataset is created.

Allocation Parameters

The log and the script datasets are created for each debugging session. **Values for the Space Units, Primary, and Secondary fields are required**. Valid default values for allocation of a log or script dataset are as follows:

Data Class	If required, enter the data class name defined by your site that contains the dataset attributes related to the allocation of the dataset. A data class is displayed only when SMS (Storage Management Subsystem) is being used.
Space Units	Valid values are TRK, CYL, or a block size (1-32760).
Primary	Valid values are 0-32760.
Secondary	Valid values are 0-32760.
Storage Class	If required, enter the storage class name defined by your site that contains the dataset attributes related to the storage occupied by the dataset. A storage class is displayed only when SMS (Storage Management Subsystem) is being used.
Unit	Enter the unit if required at your site. Otherwise, leave blank for defaults.
Volume	Enter the volume if required at your site. Otherwise, leave blank for defaults.

The log and script datasets remain open throughout the current session. If you modify the above parameters after these datasets have been allocated, the new values take effect the *next time* you begin a debugging session.

Disposition After the Test

A value is required for Process Option. The action specified is taken when you END from the test screen. Specify any of the following process options to indicate the disposition of the log or script dataset *after* the debugging session:

D Delete the dataset without printing it.

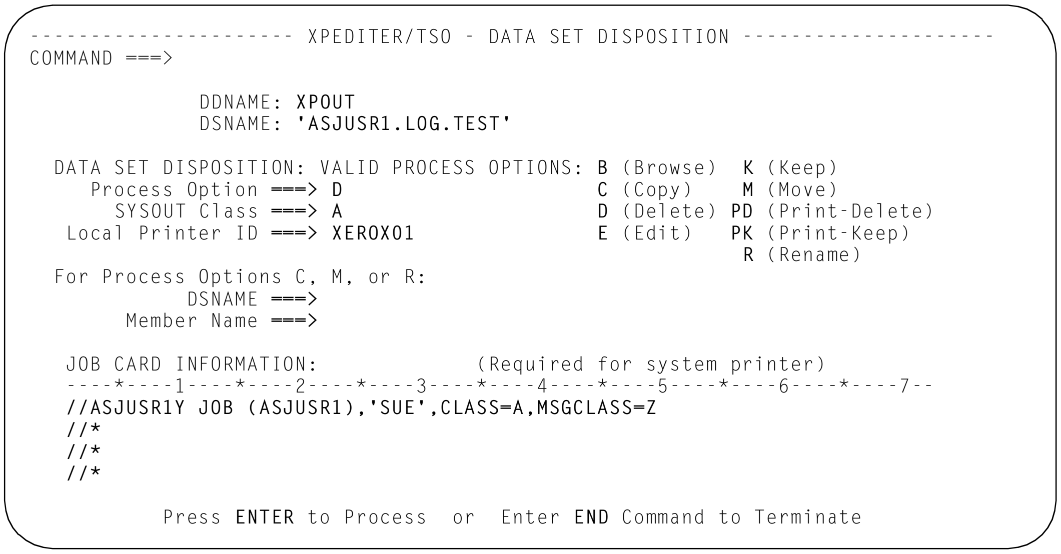
K Keep the dataset without printing it.

PD Print and delete the dataset.

PK Print and keep the dataset.

? You will be prompted to select a process option each time you exit the test screen.

Displays the Data Set Disposition screen shown in Figure C-11 after you END from the test screen so you can select a disposition process option. The dataset's dsname and the ddname to which it is allocated are prefilled on this screen.



```
--------------------- XPEDITER/TSO - DATA SET DISPOSITION --------------------
COMMAND ===>

              DDNAME: XPOUT
              DSNAME: 'ASJUSR1.LOG.TEST'

  DATA SET DISPOSITION: VALID PROCESS OPTIONS: B (Browse)  K (Keep)
     Process Option ===> D                     C (Copy)    M (Move)
       SYSOUT Class ===> A                     D (Delete) PD (Print-Delete)
   Local Printer ID ===> XEROX01               E (Edit)   PK (Print-Keep)
                                                           R (Rename)
  For Process Options C, M, or R:
             DSNAME ===>
        Member Name ===>

   JOB CARD INFORMATION:             (Required for system printer)
   ----*----1----*----2----*----3----*----4----*----5----*----6----*----7--
   //ASJUSR1Y JOB (ASJUSR1),'SUE',CLASS=A,MSGCLASS=Z
   //*
   //*
   //*

           Press ENTER to Process  or  Enter END Command to Terminate
```

Figure C-11. Data Set Disposition Screen Displayed After the Debugging Session is Ended

The fields on the screen shown in Figure C-11 are:

Process Option
Any of the following process options can be specified:

B Browse the log or script dataset.

C Copy the log or script dataset to the partitioned dataset named in the DSNAME field. The PDS member name must be entered in the Member Name field.

D Delete the log or script dataset without printing it.

E Display the log or script dataset so you can edit it before copying it to another dataset.

K Keep the log or script dataset without printing it. The file is saved as a sequential dataset by the same name.

M Move the log or script dataset to the partitioned dataset named in the DSNAME field. The PDS member name must be entered in the Member Name field.

PD Print and delete the log or script dataset.

PK Print and keep the log or script dataset.

R Rename a sequential dataset and enter its new name in the DSNAME field. No member name is required because only a sequential dataset can be specified for the Rename option.

SYSOUT Class
A default SYSOUT class can be entered for XPEDITER/TSO datasets. If specified, it is used when process option PD (print and delete) or PK (print and keep) is entered. The SYSOUT class can be A-Z, 0-9, or *.

If the SYSOUT class is to be used to direct printing to the appropriate output device, JOB CARD INFORMATION must also be specified.

Local Printer ID
A default local printer ID can be specified for the XPEDITER/TSO dataset. If specified, it is used when process option PD (print and delete) or PK (print and keep) is selected. The local printer ID is the name your installation assigned to a local IBM 328x printer (for example, XEROX01, as shown in Figure C-11 on page C-10). For local printing, the TSO command processor DSPRINT must be installed on the Local Printer Support installation screen.

DSNAME
Required for process options C, M, and R. Enter the name of the partitioned dataset into which the log or script dataset is to be copied, moved, or renamed.

Member Name
Required for process options C and M. Enter the name of the PDS member.

JOB CARD INFORMATION
Required if SYSOUT Class is specified. Enter up to four default job statements to submit a background job to print the log or script dataset when the current debugging session ends.

The remaining field on the Log Dataset screen (Figure C-10) is:

Disposition Before the Test
A value for Process Option is required only if a DSNAME is specified. Use any of the following process options to indicate the disposition of an existing log or script dataset *before* the debugging session begins:

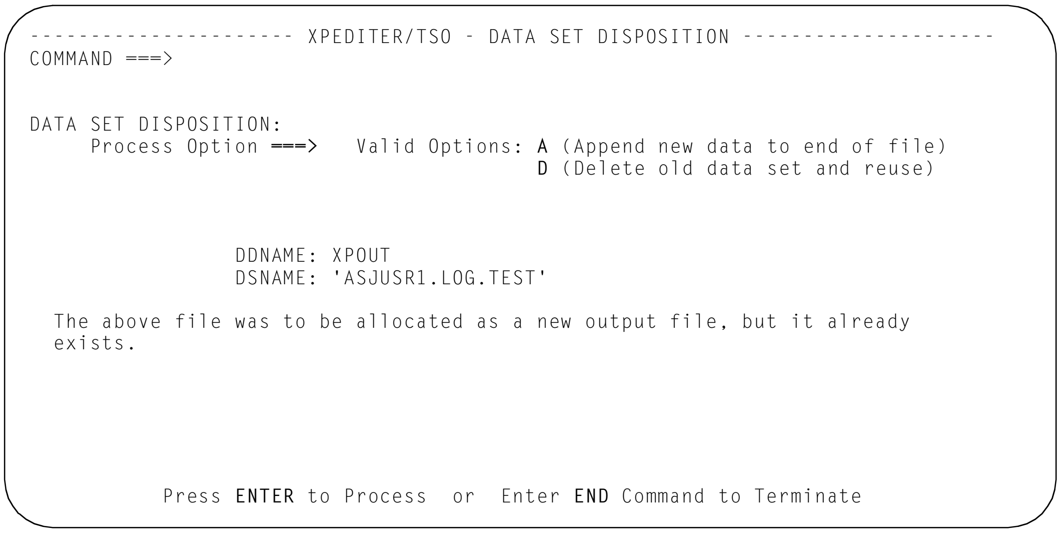
A Append the new dataset to the end of the old dataset.

D Delete the old dataset and reuse the same DSNAME.

? Displays the Data Set Disposition screen shown in Figure C-12 before the beginning of a debugging session so you can specify whether the new dataset is to be appended (option A) to the end of the existing log or script dataset or to delete (option D) the existing dataset before creating the log or script dataset for this debugging session.

Note: Attempting to allocate a dataset with the same name as an existing dataset causes a file allocation error. This Data Set Disposition screen lets you specify what to do when that situation occurs.

The DDNAME field on the Data Set Disposition screen contains the ddname to which this dataset is about to be allocated. For example, the log dataset is always allocated to the XPOUT ddname. The dsname you entered on the previous screen is prefilled in the DSNAME field.



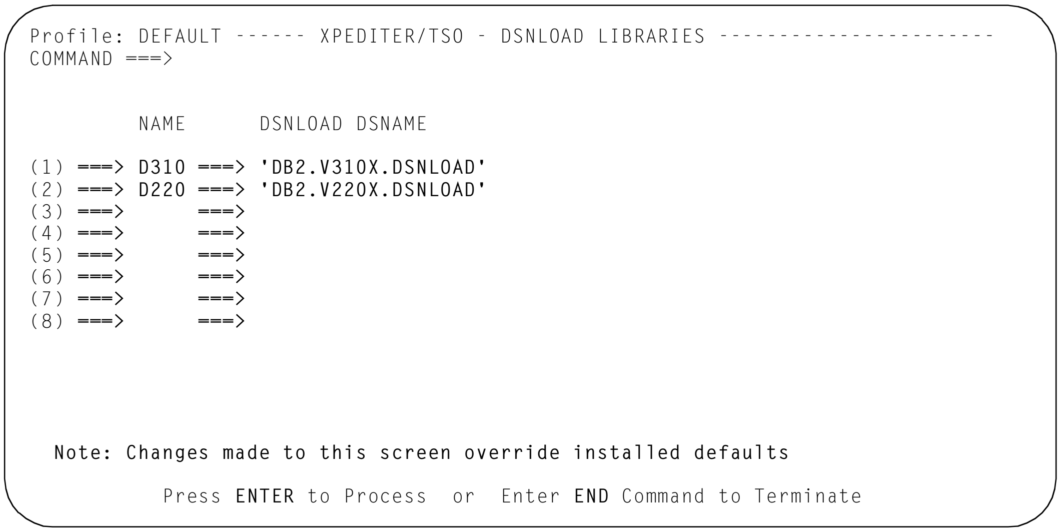
```
--------------------- XPEDITER/TSO - DATA SET DISPOSITION --------------------
COMMAND ===>


DATA SET DISPOSITION:
     Process Option ===>   Valid Options: A (Append new data to end of file)
                                          D (Delete old data set and reuse)



                 DDNAME: XPOUT
                 DSNAME: 'ASJUSR1.LOG.TEST'

  The above file was to be allocated as a new output file, but it already
  exists.





           Press ENTER to Process  or  Enter END Command to Terminate
```

Figure C-12. Data Set Disposition Screen Displayed Before the Debugging Session Begins

DB2 System Name and DSNLOAD Libraries Screen

Establishing the connection to DB2 requires a DB2 system ID and access to the DB2 programs that reside in the DSNLOAD dataset that was created when DB2 was installed. The DB2 system name can be defaulted, and the DSNLOAD dataset(s) can be placed in the installation LPALIB or LINKLIST. The DB2 System Name and DSNLOAD screen is used to define required or additional DB2 system IDs and DSNLOAD datasets for access to DB2.

Access to multiple DB2 systems and multiple DSNLOAD datasets is controlled by the entries on this screen. If your site defined a default DB2 system ID and the DB2 DSNLOAD dataset is permanently allocated when XPEDITER/TSO is activated, then this screen can be left blank.



```
Profile: DEFAULT ------ XPEDITER/TSO - DSNLOAD LIBRARIES ----------------------
COMMAND ===>


         NAME      DSNLOAD DSNAME

(1) ===> D310 ===> 'DB2.V310X.DSNLOAD'
(2) ===> D220 ===> 'DB2.V220X.DSNLOAD'
(3) ===>      ===>
(4) ===>      ===>
(5) ===>      ===>
(6) ===>      ===>
(7) ===>      ===>
(8) ===>      ===>






  Note: Changes made to this screen override installed defaults
           Press ENTER to Process  or  Enter END Command to Terminate
```

Figure C-13. DSNLOAD Libraries Screen

The fields on the DSNLOAD Libraries screen are described below:

NAME
: Defines DB2 system ID names that can be used at your site to connect to a DB2 system. DSNLOAD datasets can also be associated with the DB2 system IDs on this screen. If you use more than one DB2 system, then each should be defined on this screen.

For standard debugging sessions, specify a specific DB2 system ID to be used when activating DB2. This value becomes the system parameter option for the DSN command, and the associated DSNLOAD datasets are included in XTASKLIB.

For dialog debugging sessions, the DSNLOAD datasets are included in XTASKLIB, but you must issue the DSN if it is required.

For IMS and BTS debugging sessions, the DSNLOAD datasets are allocated to DFSESL, as well as XTASKLIB. IMS also connects through the IMS/DB2 interface module DSNMTV01.

DSNLOAD NAME
Associates the DB2 system ID name with the DSNLOAD datasets. For each name, one or more DSNLOAD datasets can be defined. The following example shows four possible DB2 connections with various DSNLOAD dataset combinations:

```
                NAME        DSNLOAD DSNAME
      (1) ===> DSNX   ===> 'DSNX.LOADLIB'
      (2) ===> DSNX   ===> 'DSNX.RUNLIB'
      (3) ===> DSNY   ===>
      (4) ===> DSNZ   ===> 'DSNZ.LOADLIB'
      (5) ===>        ===> 'DSNY.LOADLIB'
```

The DB2 system identified by DSNX has two DSNLOAD datasets allocated to XTASKLIB.

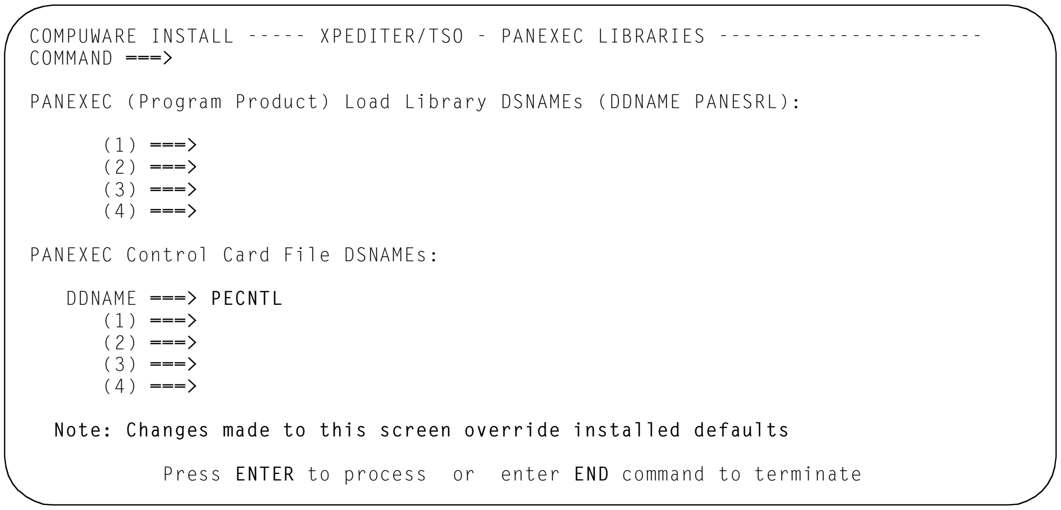
The DB2 system identified by DSNZ has one DSNLOAD dataset allocated.

The system identified by DSNY does not have any DSNLOAD datasets allocated.

A DB2 test with no system specified is allocated on the DSNLOAD dataset.

PANEXEC Libraries Screen

The PANEXEC Libraries screen is used to enter the library names and control card file datasets necessary to debug your application programs using PANEXEC.



```
COMPUWARE INSTALL ----- XPEDITER/TSO - PANEXEC LIBRARIES ----------------------
COMMAND ===>

PANEXEC (Program Product) Load Library DSNAMEs (DDNAME PANESRL):

      (1) ===>
      (2) ===>
      (3) ===>
      (4) ===>

PANEXEC Control Card File DSNAMEs:

   DDNAME ===> PECNTL
      (1) ===>
      (2) ===>
      (3) ===>
      (4) ===>

  Note: Changes made to this screen override installed defaults

           Press ENTER to process  or  enter END command to terminate
```

Figure C-14. PANEXEC Libraries Screen

The fields on the PANEXEC Libraries screen are:

PANEXEC Load Library DSNAMEs
Enter the dataset name normally allocated to ddname PANESRL.

PANEXEC Control Card File DSNAMES
Enter the default ddname associated with your PANEXEC control cards. The default is PECNTL.

If you normally direct your PECNTL (or site default PANEXEC control cards ddname) to a dataset rather than to instream data, enter that dataset in the next field.

IMS Setup Menu

The menu that appears for option I (IMS) on the IMS Test Setup Menu is shown in Figure C-15. The screens for these options are used to specify the datasets to be allocated to the IMS ddnames. The nine screens can be selected individually or all together as a complete sequence.

The following screens are required:

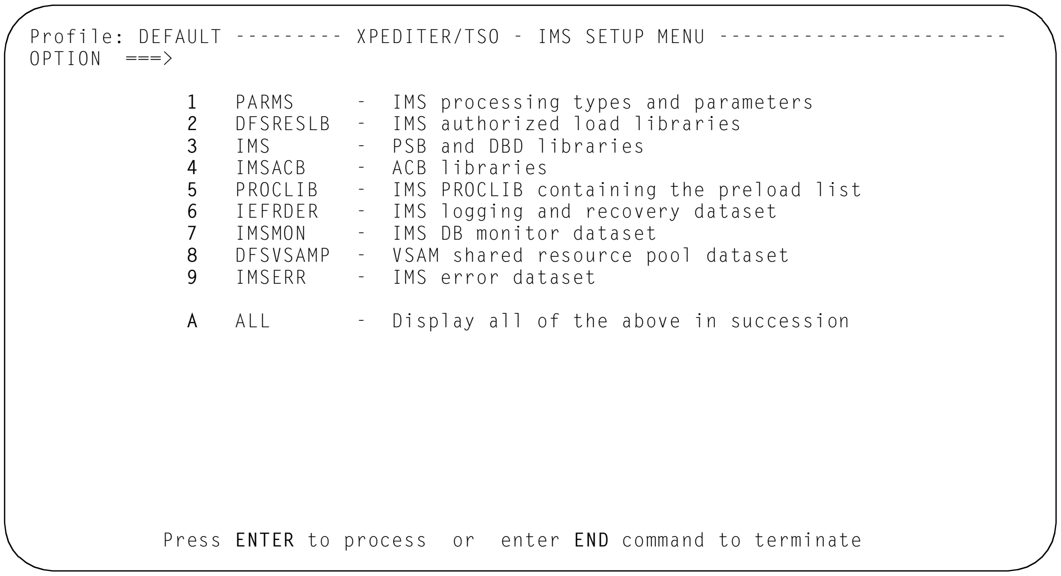
- IMS processing types and parameters (option 1)
- IMS authorized load libraries (option 2)
- PSB and DBD libraries (option 3)
- ACB libraries (option 4)

All these parameters are required to debug an IMS program.

The rest of the selections on the menu are optional.

The values specific to each of the screens should have been completed at installation time. You may want to merely verify them, or in some cases, add to or override the installation defaults. For example, you may want to change program parameters specific to your application.

All the IMS setup screens are described in the following subsections.

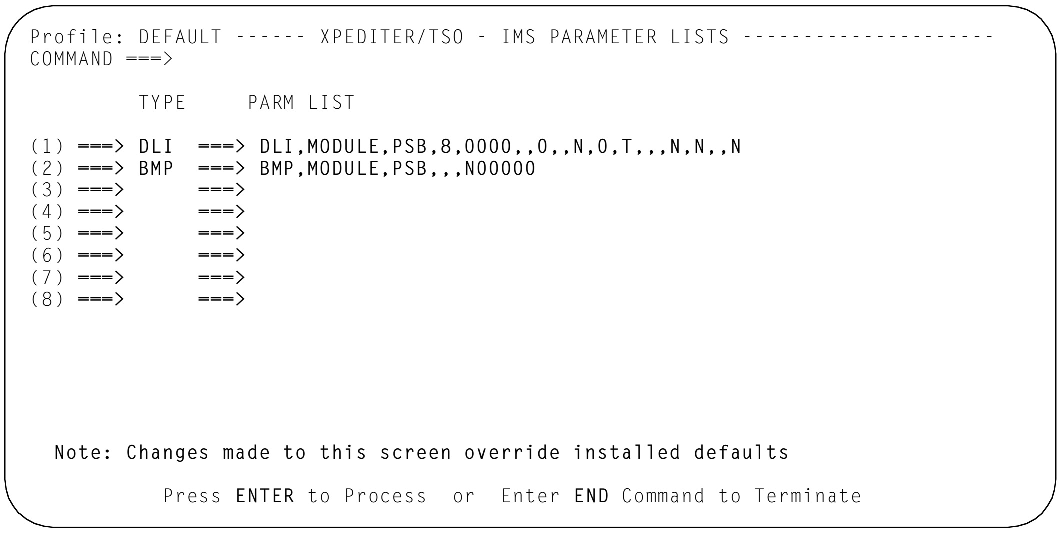


```
Profile: DEFAULT --------- XPEDITER/TSO - IMS SETUP MENU -----------------------
OPTION  ===>

             1   PARMS     -  IMS processing types and parameters
             2   DFSRESLB  -  IMS authorized load libraries
             3   IMS       -  PSB and DBD libraries
             4   IMSACB    -  ACB libraries
             5   PROCLIB   -  IMS PROCLIB containing the preload list
             6   IEFRDER   -  IMS logging and recovery dataset
             7   IMSMON    -  IMS DB monitor dataset
             8   DFSVSAMP  -  VSAM shared resource pool dataset
             9   IMSERR    -  IMS error dataset

             A   ALL       -  Display all of the above in succession







           Press ENTER to process  or  enter END command to terminate
```

Figure C-15. IMS Setup Menu Screen

IMS Parameter Lists Screen

The IMS Parameter Lists screen lets you specify the valid program types and their corresponding parameter lists.



```
Profile: DEFAULT ------ XPEDITER/TSO - IMS PARAMETER LISTS --------------------
COMMAND ===>

         TYPE     PARM LIST

(1) ===> DLI  ===> DLI,MODULE,PSB,8,0000,,0,,N,0,T,,,N,N,,N
(2) ===> BMP  ===> BMP,MODULE,PSB,,,N00000
(3) ===>      ===>
(4) ===>      ===>
(5) ===>      ===>
(6) ===>      ===>
(7) ===>      ===>
(8) ===>      ===>




  Note: Changes made to this screen override installed defaults

           Press ENTER to Process  or  Enter END Command to Terminate
```

Figure C-16. IMS Parameter Lists Screen

The fields on the IMS Parameter Lists screen are:

TYPE
Enter the IMS program types supported at your site (DLI, DBB, BMP, etc.). Create a separate line for each type and its corresponding PARM list.

PARM LIST
Enter the PARM list for the specified type. Each list contains the program type followed by the keywords ",MODULE,PSB," and the rest of the parameters used at your site. At execution time, the ",MODULE,PSB," string is replaced with the program name and program specification block that the user enters on the test screen.

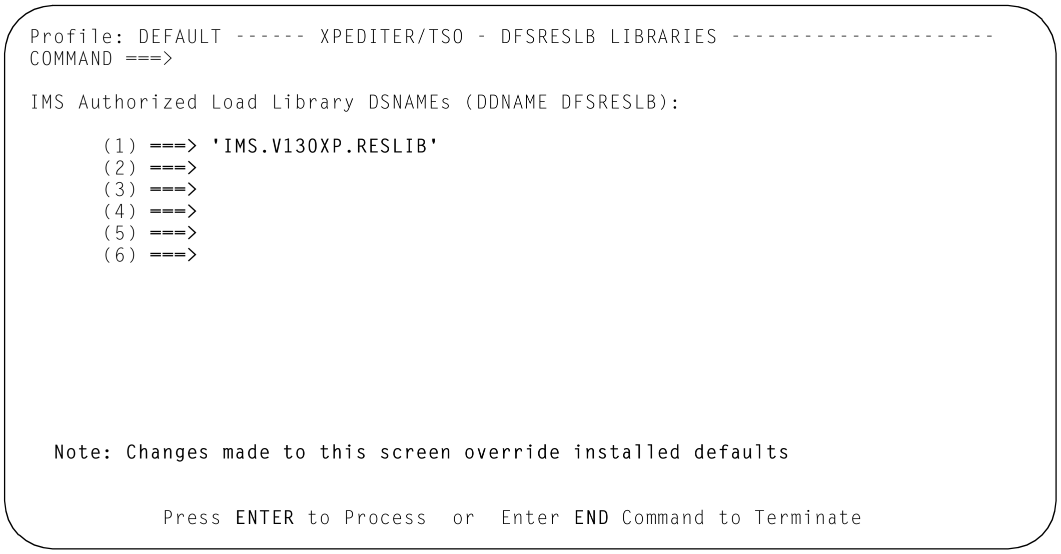
The supplied default parameter string(s) probably will not function correctly at your site. If you are uncertain what parameter list to specify, consult one of the following information sources:

- One of your site's typical DLI/BMP batch PROCs
- Your system programmer
- IBM's *IMS/VS System Programming Reference Manual*
- General information available through the HELP screens

Note: Special characters (anything other than numerics, alphabetic characters, $, #, or @) must be enclosed in quotation marks.

IMS Load (DFSRESLB) Libraries Screen

The IMS DFSRESLB Libraries screen is used to allocate the dsnames of the IMS authorized load libraries to the DFSRESLB ddname. The DFSRESLB dataset contains all the system load modules that make up the IMS software, including both DL/I and the data communications component.



```
Profile: DEFAULT ------ XPEDITER/TSO - DFSRESLB LIBRARIES ----------------------
COMMAND ===>

IMS Authorized Load Library DSNAMEs (DDNAME DFSRESLB):

      (1) ===> 'IMS.V130XP.RESLIB'
      (2) ===>
      (3) ===>
      (4) ===>
      (5) ===>
      (6) ===>







  Note: Changes made to this screen override installed defaults


           Press ENTER to Process  or  Enter END Command to Terminate
```

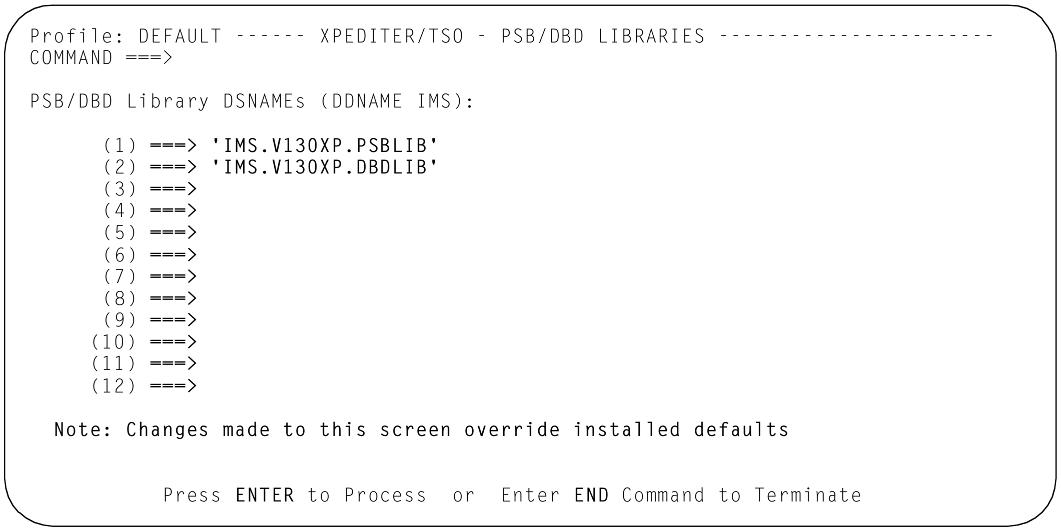
Figure C-17. IMS DFSRESLB Libraries Screen

The field on this screen is:

IMS Authorized Load Library DSNAMEs
Enter from one to six dsnames. Consult a batch IMS PROC at your site if you are not sure which dsnames to enter.

PSB and DBD Libraries Screen

The PSB/DBD Libraries screen (Figure C-18) is used to list your IMS PSB and DBD library dataset names. The PSBLIB contains your program specification blocks (PSBs). The DBDLIB contains your data base definitions (DBDs). Both of these datasets are required with DB or DB/DC systems. They can also be used if a DL/I batch region is being executed.



```
Profile: DEFAULT ------ XPEDITER/TSO - PSB/DBD LIBRARIES -----------------------
COMMAND ===>

PSB/DBD Library DSNAMEs (DDNAME IMS):

      (1) ===> 'IMS.V130XP.PSBLIB'
      (2) ===> 'IMS.V130XP.DBDLIB'
      (3) ===>
      (4) ===>
      (5) ===>
      (6) ===>
      (7) ===>
      (8) ===>
      (9) ===>
     (10) ===>
     (11) ===>
     (12) ===>

  Note: Changes made to this screen override installed defaults


           Press ENTER to Process  or  Enter END Command to Terminate
```

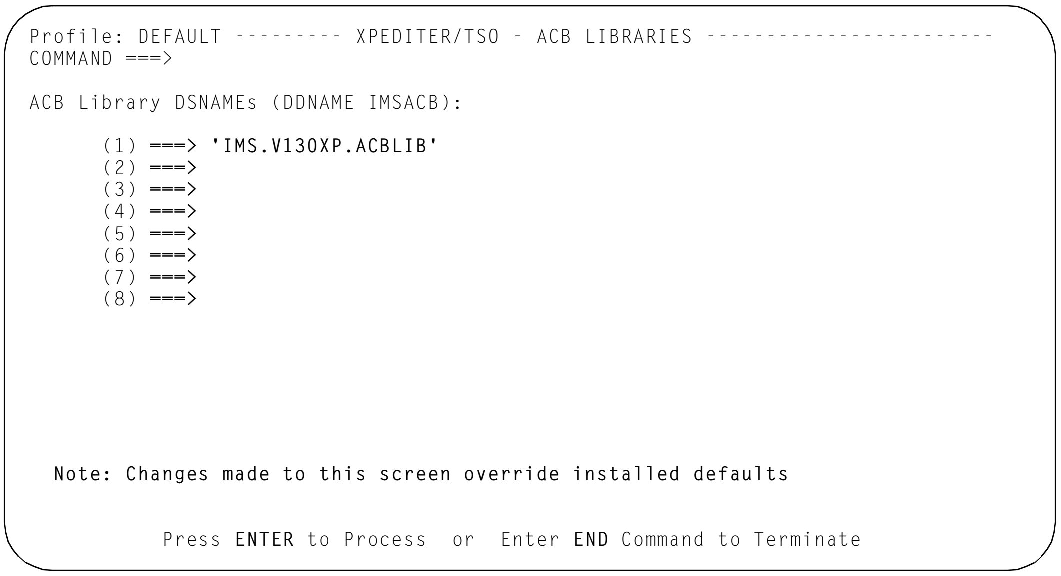
Figure C-18. PSB/DBD Libraries Screen

The field on this screen is:

PSB/DBD Library DSNAMEs
Enter from zero to twelve dsnames. Consult a batch IMS PROC at your site if you are unsure of the dsnames. If the application being debugged uses ACB libraries, the DSNAME field can be left blank because the ACBLIB dataset stores the combined DBDs and PSBs.

ACB Libraries Screen

The ACB Libraries screen is used to enter your ACB library dataset names. When an IMS application program is executed, IMS must combine the information in the DBD and PSB before the program can be executed. Your shop can merge the DBD and PSB information in the ACBLIB, so that IMS will not have to do this each time a program runs. The use of the ACBLIB dataset is optional for batch IMS programs, but is required for MPP and BMP programs.



```
Profile: DEFAULT --------- XPEDITER/TSO - ACB LIBRARIES -----------------------
COMMAND ===>

ACB Library DSNAMEs (DDNAME IMSACB):

      (1) ===> 'IMS.V130XP.ACBLIB'
      (2) ===>
      (3) ===>
      (4) ===>
      (5) ===>
      (6) ===>
      (7) ===>
      (8) ===>






  Note: Changes made to this screen override installed defaults


           Press ENTER to Process  or  Enter END Command to Terminate
```

Figure C-19. ACB Libraries Screen

The field on this screen is:

ACB Library DSNAMEs
: Enter from zero to eight dsnames. Consult a batch PROC at your site if you are unsure of the dsnames. The DSNAME field can be left blank if the application being debugged uses PSB and DBD libraries.

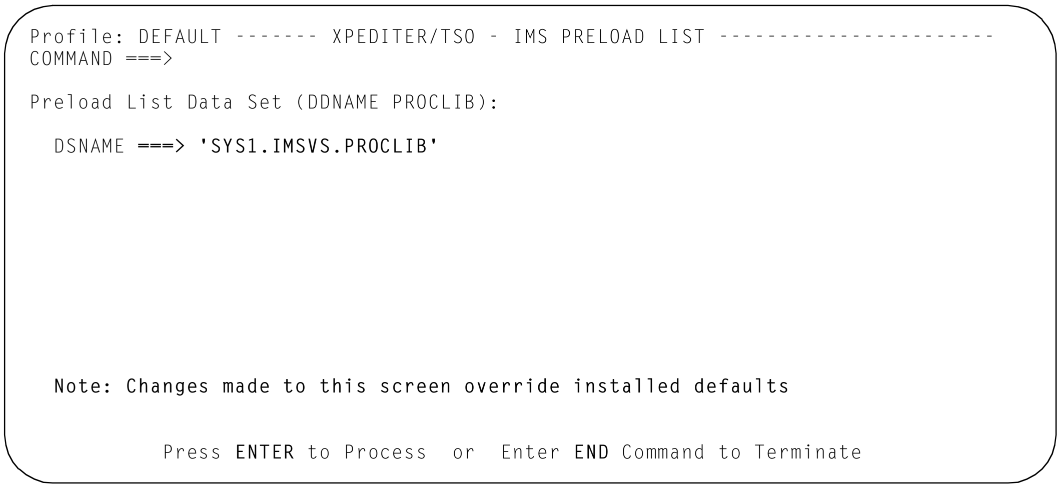
IMS Preload List Screen

The IMS Preload List screen lets you enter the PROCLIB dataset name. Both IMS and BTS use the PROCLIB ddname to point to the dataset where preload specifications are stored. The IMSVS.PROCLIB dataset stores all IMS-generated cataloged procedures and jobs. DFSMPL*xx* is the IMSVS.PROCLIB member that contains the preload list, which is a group of modules to be loaded before IMS loads the user's program.

A 2-character suffix from a parameter list is used to select the preload list to be executed. For example, suppose you specify that the following DLI parameter list is to be used with your program. (The DLI parameter list shown below would be displayed on your IMS Parameter Lists screen.)

```
       TYPE      PARM LIST
  ===> DLI   ===> DLI,MODULE,PSB,12,0000,00,0,,N
```

Notice that the sixth parameter is a pair of zeros. This suffix is concatenated with DFSMPL*xx* to define DFSMPL00 as the name of your preload list. Omit the suffix if no preload list is needed.



```
Profile: DEFAULT ------- XPEDITER/TSO - IMS PRELOAD LIST ----------------------
COMMAND ===>

Preload List Data Set (DDNAME PROCLIB):

  DSNAME ===> 'SYS1.IMSVS.PROCLIB'










  Note: Changes made to this screen override installed defaults


           Press ENTER to Process  or  Enter END Command to Terminate
```

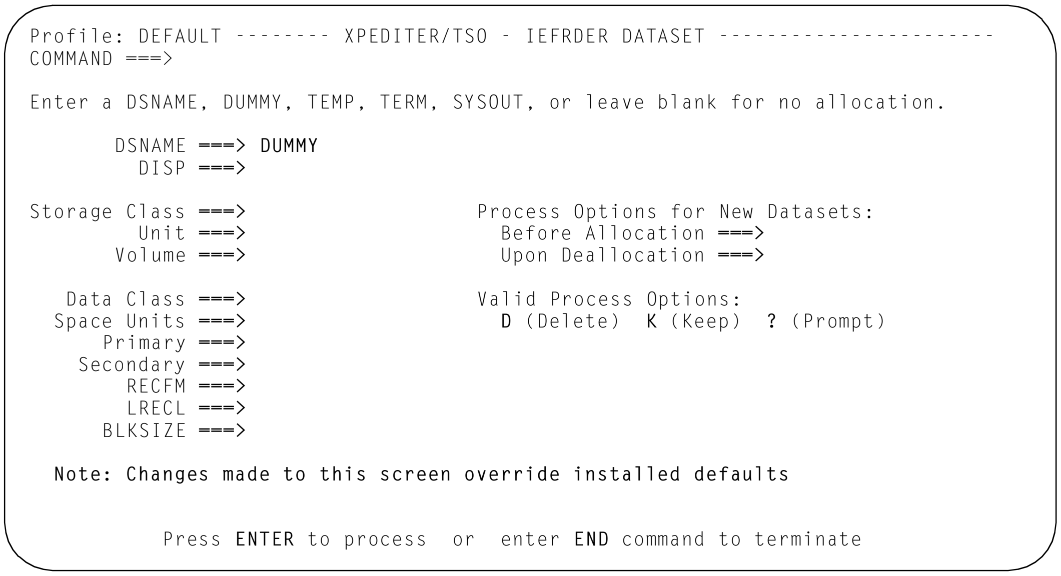
Figure C-20. IMS Preload List Screen

The field on this screen is:

Preload List Data Set
Enter the name of the dataset containing preload lists, **not** the name of the dataset where XPEDITER/TSO or IMS PROCs are stored, unless the two are identical. Consult an IMS batch PROC at your site if you are not sure which dsname to enter. If this ddname is not required, leave the DSNAME field blank.

Logging and Recovery (IEFRDER) Dataset Screen

The IEFRDER Dataset screen is used to allocate a dataset to the IEFRDER ddname. Supplying an IEFRDER dataset causes the IMS logging facility to be invoked.



```
Profile: DEFAULT -------- XPEDITER/TSO - IEFRDER DATASET ----------------------
COMMAND ===>

Enter a DSNAME, DUMMY, TEMP, TERM, SYSOUT, or leave blank for no allocation.

       DSNAME ===> DUMMY
         DISP ===>

Storage Class ===>                  Process Options for New Datasets:
         Unit ===>                    Before Allocation ===>
       Volume ===>                    Upon Deallocation ===>

   Data Class ===>                  Valid Process Options:
  Space Units ===>                    D (Delete)  K (Keep)  ? (Prompt)
      Primary ===>
    Secondary ===>
        RECFM ===>
        LRECL ===>
      BLKSIZE ===>

  Note: Changes made to this screen override installed defaults


           Press ENTER to process  or  enter END command to terminate
```

Figure C-21. IEFRDER Dataset Screen

The fields on this screen are:

DSNAME
Enter a qualified dsname (enclosed in quotes), DUMMY, TEMP, TERM, or SYSOUT (JES dataset), or leave this field blank. If the dsname is entered without quotes, the user ID (or other dsname high-level qualifier) is prefixed and quotes are added. If the DSNAME field is left blank, the dataset is not allocated.

Consult a batch IMS PROC at your site if you are not sure which dsname to enter. If the IEFRDER ddname is not required, you can leave the DSNAME field blank. Actually, most sites allocate this ddname to DUMMY for debugging rather than leaving it blank. This conforms to common batch and debugging message region usage.

DISP

Valid values for the DISP field are:

S	Shared (SHR)
M	Modified (MOD)
O	Old
N	New

If you are allocating an existing dataset, you can specify a disposition of SHR, OLD, or MOD. If you do not enter a value in the DISP field, a SHR disposition is assumed. Datasets with a disposition of SHR can be concatenated.

If the DSNAME field contains a value of DUMMY, TEMP, TERM, or SYSOUT, leave the DISP field blank.

Enter a disposition of NEW for a new or temporary dataset. If you are allocating a new (DSNAME='qualified-dsname') or temporary file (DSNAME=TEMP), you must enter values for all the fields below except Volume and Unit, which are optional for a NEW or TEMP file.

If you specify a new dataset, it is allocated before your debugging session begins and is deallocated immediately after your debugging session ends. The fields for the Process Options for New Datasets let you specify how the new dataset is to be allocated and deallocated.

For more information about NEW, OLD, MOD, or SHR dataset dispositions, refer to "Types of Files That Can Be Allocated" on page A-16.

Storage Class

Enter the storage class name defined by your site that contains the dataset attributes related to the storage occupied by the dataset. A storage class is displayed only when SMS (Storage Management Subsystem) is being used.

Unit

If your site requires it, you must supply a value for the Unit field. Otherwise, leave blank for the default value.

Volume

If your site requires it, you must supply a value for the Volume field. Also enter the volume if you are specifying an uncataloged dataset. Otherwise, leave blank for the default value.

Data Class

Enter the data class name defined by your site that contains the dataset attributes related to the allocation of the dataset. A data class is displayed only when SMS (Storage Management Subsystem) is being used.

Space Units

Valid values are TRK, CYL, or a block size (1-32760).

Primary

Valid values are 0-32760.

Secondary

Valid values are 0-32760.

RECFM

Valid values are FBM, FBA, FB, F, VBM, VBA, VBS, VS, VB, V, and U.

LRECL

Valid values are 0-32760. Omit this field if RECFM is U for undefined length records.

BLKSIZE
Valid values are 0-32760.

If you are allocating a DUMMY or SYSOUT file (DSNAME=DUMMY or DSNAME=SYSOUT), you must enter the block size.

Before Allocation
Valid process options are:

D Automatically delete the old file, if one exists, before a new dataset is allocated.

K Keep the old dataset.

? Display the Data Set Disposition screen each time a dataset is to be allocated, so that you can decide whether to delete or keep the old one.

Upon Deallocation
Valid process options are:

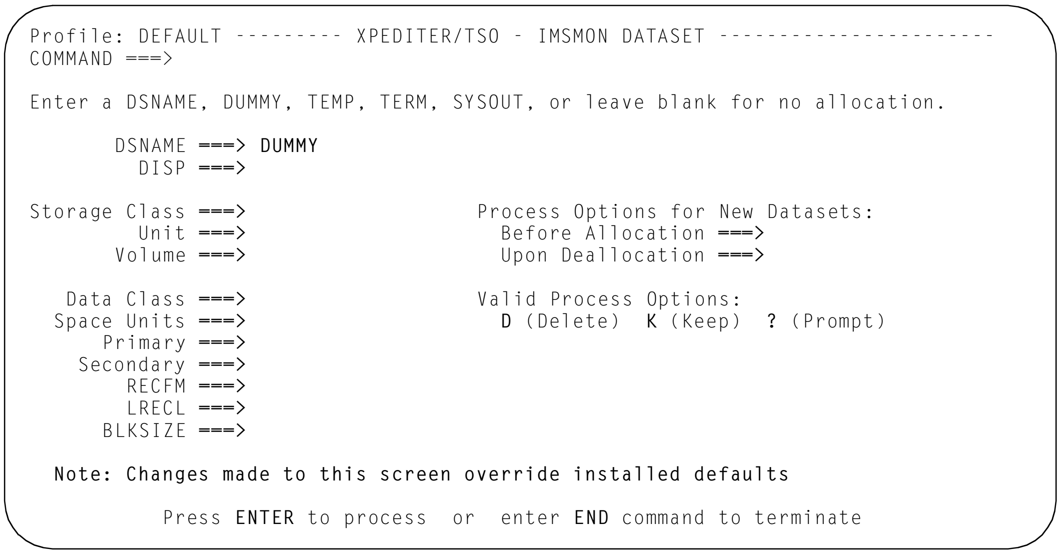
D Automatically delete the new file after every debugging session.

K Always keep the new dataset.

? Display the Data Set Disposition screen each time a dataset is to be deallocated, so that you can decide whether to delete or keep the new one.

IMS Monitor (IMSMON) Dataset Screen

The IMSMON Dataset screen is used to allocate the DB and DC monitor datasets to the IMSMON ddname. IMS run-time activities, especially database calls, are recorded in the IMS monitor file. This dataset can be used as an input for report programs that list, summarize, or analyze IMS activities.



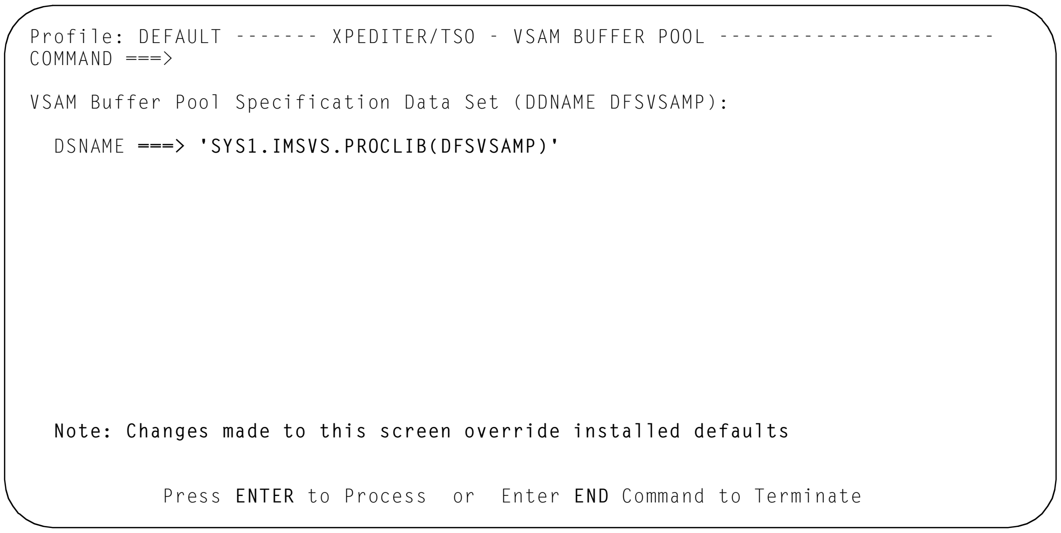
```
Profile: DEFAULT --------- XPEDITER/TSO - IMSMON DATASET ----------------------
COMMAND ===>

Enter a DSNAME, DUMMY, TEMP, TERM, SYSOUT, or leave blank for no allocation.

       DSNAME ===> DUMMY
         DISP ===>

Storage Class ===>                  Process Options for New Datasets:
         Unit ===>                    Before Allocation ===>
       Volume ===>                    Upon Deallocation ===>

   Data Class ===>                  Valid Process Options:
  Space Units ===>                    D (Delete)  K (Keep)  ? (Prompt)
      Primary ===>
    Secondary ===>
        RECFM ===>
        LRECL ===>
      BLKSIZE ===>

  Note: Changes made to this screen override installed defaults

           Press ENTER to process  or  enter END command to terminate
```

Figure C-22. IMSMON Dataset Screen

The valid values for the fields on this screen are the same as for the IEFRDER screen. Refer to "Logging and Recovery (IEFRDER) Dataset Screen" on page C-19 for descriptions of these fields.

VSAM Buffer Pool Screen

The VSAM Buffer Pool screen, shown in Figure C-23, is used to enter the name of the dataset that contains the buffer length and the number of buffers to be used for a program.



```
Profile: DEFAULT ------- XPEDITER/TSO - VSAM BUFFER POOL ----------------------
COMMAND ===>

VSAM Buffer Pool Specification Data Set (DDNAME DFSVSAMP):

  DSNAME ===> 'SYS1.IMSVS.PROCLIB(DFSVSAMP)'










  Note: Changes made to this screen override installed defaults


           Press ENTER to Process  or  Enter END Command to Terminate
```

Figure C-23. VSAM Buffer Pool Screen

The field on this screen is:

VSAM Buffer Pool Specification Data Set

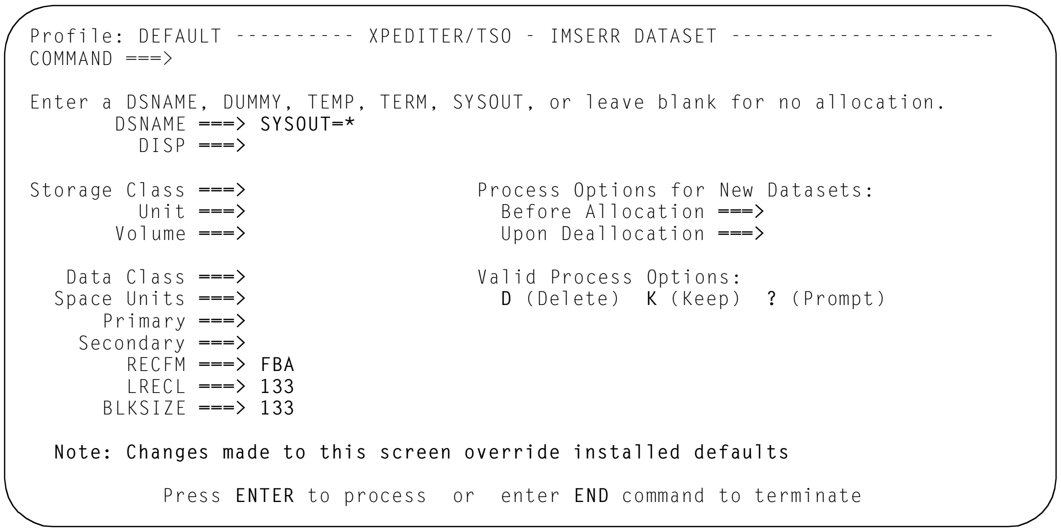
The DFSVSAMP dataset contains the buffer length and the number of buffers to be used for a program. If not stated, IMS uses a default buffer size. If the DFSVSAMP dataset is a PDS, enter its member name.

If VSAM databases are used, this dataset must be allocated. Since BTS can attach IMS more than once, define the DFSVSAMP file as a DASD dataset so it can be reread. The DFSVSAMP file can also be used for ISAM/OSAM databases.

Consult a batch PROC at your site if you are unsure of the dsname. If this ddname is not required, leave the DSNAME field blank.

IMS Error (IMSERR) Dataset Screen

The IMSERR Dataset screen is used to allocate datasets to the IMSERR ddname. An IMSERR dataset contains the formatted dump of the IMS/VS GSAM control blocks when an error occurs, or when a DUMP or SNAP call is issued to a GSAM PCB.



```
Profile: DEFAULT ---------- XPEDITER/TSO - IMSERR DATASET ---------------------
COMMAND ===>

Enter a DSNAME, DUMMY, TEMP, TERM, SYSOUT, or leave blank for no allocation.
       DSNAME ===> SYSOUT=*
         DISP ===>

Storage Class ===>                  Process Options for New Datasets:
         Unit ===>                    Before Allocation ===>
       Volume ===>                    Upon Deallocation ===>

   Data Class ===>                  Valid Process Options:
  Space Units ===>                    D (Delete)  K (Keep)  ? (Prompt)
      Primary ===>
    Secondary ===>
        RECFM ===> FBA
        LRECL ===> 133
      BLKSIZE ===> 133

  Note: Changes made to this screen override installed defaults

           Press ENTER to process  or  enter END command to terminate
```

Figure C-24. IMSERR Dataset Screen

The valid values for the fields on this screen are the same as for the IEFRDER screen. Refer to "Logging and Recovery (IEFRDER) Dataset Screen" on page C-19 for descriptions of these fields.

Note: Consult a batch IMS PROC at your site if you are unsure of the dsname. Most sites allocate this ddname to DUMMY for debugging.

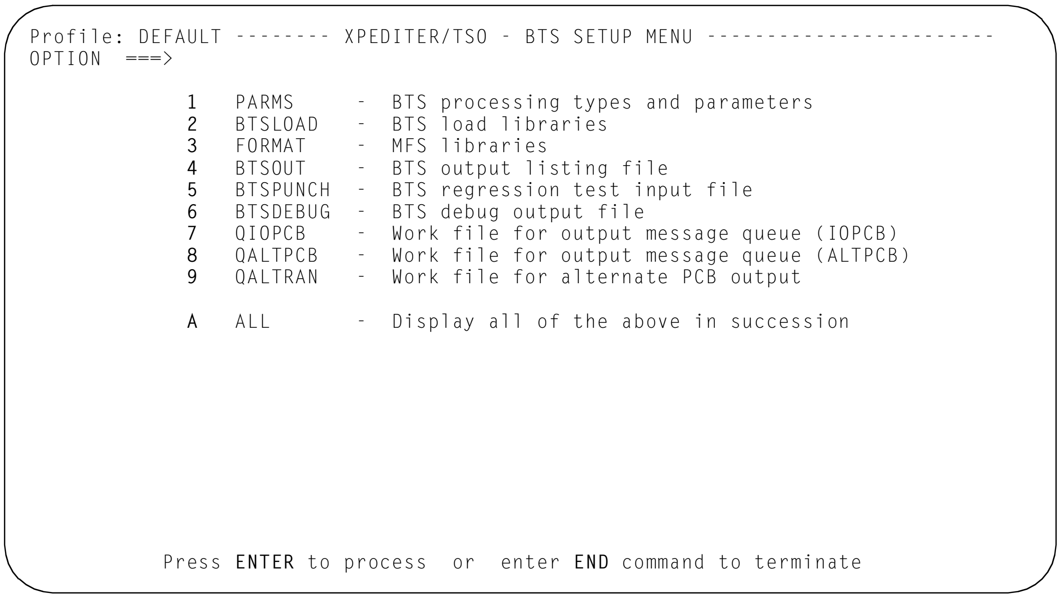
BTS Setup Menu

The menu that appears for option B (BTS) on the BTS Test Setup Menu is shown in Figure C-25. The BTS Setup Menu is used to select screens where you can list the datasets to be allocated to BTS ddnames. The values specific to each of the option screens should have been completed at installation time. You may want to verify them, or in some cases, add to or override the installation defaults. The nine screens can be selected individually or all together as a complete sequence.

The following parameters are required to debug a BTS program:

1. BTS processing types and parameters (option 1)
2. BTS load libraries (option 2)
3. BTSIN dataset, which is dynamically allocated by the application programmer during the debugging session

All of the BTS setup screens are described in the following subsections.

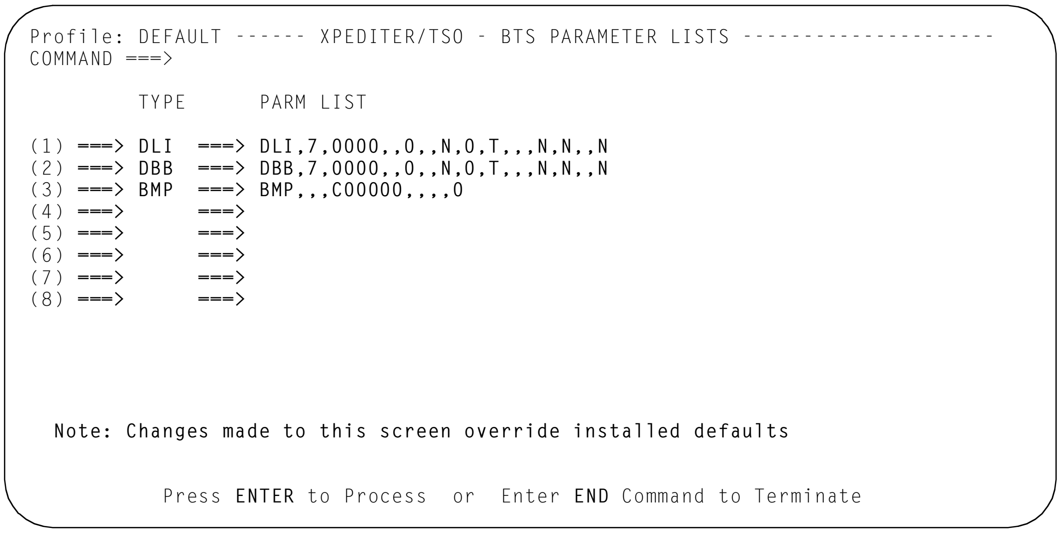


```
Profile: DEFAULT -------- XPEDITER/TSO - BTS SETUP MENU -----------------------
OPTION  ===>

             1   PARMS      -  BTS processing types and parameters
             2   BTSLOAD    -  BTS load libraries
             3   FORMAT     -  MFS libraries
             4   BTSOUT     -  BTS output listing file
             5   BTSPUNCH   -  BTS regression test input file
             6   BTSDEBUG   -  BTS debug output file
             7   QIOPCB     -  Work file for output message queue (IOPCB)
             8   QALTPCB    -  Work file for output message queue (ALTPCB)
             9   QALTRAN    -  Work file for alternate PCB output

             A   ALL        -  Display all of the above in succession










          Press ENTER to process  or  enter END command to terminate
```

Figure C-25. BTS Setup Screen

BTS Parameter Lists Screen

The BTS Parameter Lists screen is used to specify the valid program types and their corresponding PARM lists.



```
Profile: DEFAULT ------ XPEDITER/TSO - BTS PARAMETER LISTS --------------------
COMMAND ===>

         TYPE      PARM LIST

(1) ===> DLI  ===> DLI,7,0000,,0,,N,0,T,,,N,N,,N
(2) ===> DBB  ===> DBB,7,0000,,0,,N,0,T,,,N,N,,N
(3) ===> BMP  ===> BMP,,,C00000,,,,0
(4) ===>      ===>
(5) ===>      ===>
(6) ===>      ===>
(7) ===>      ===>
(8) ===>      ===>




  Note: Changes made to this screen override installed defaults


           Press ENTER to Process  or  Enter END Command to Terminate
```

Figure C-26. BTS Parameter Lists Screen

The fields on this screen are:

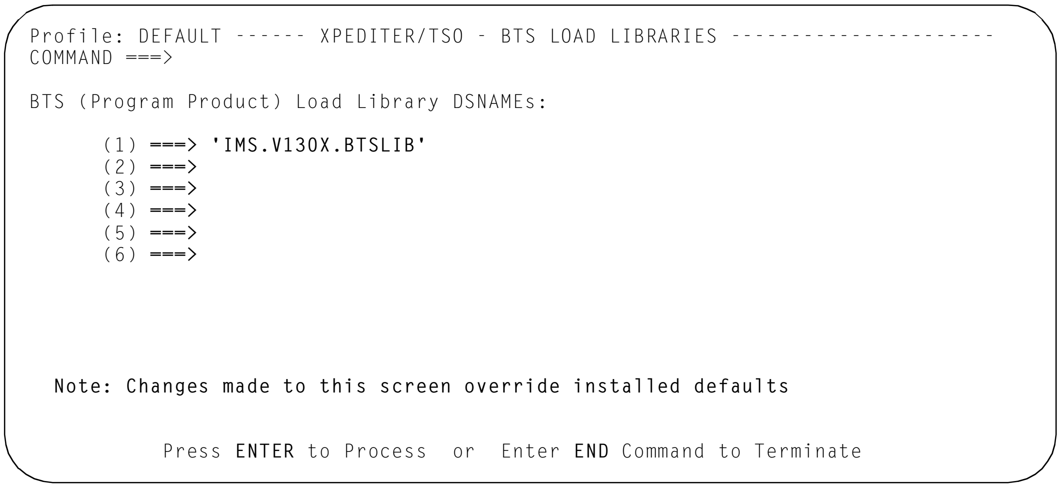
TYPE
Enter the BTS program types supported at your site (DLI, DBB, BMP, etc.). Create a separate line for each type and its corresponding PARM list.

PARM LIST
Enter the parameter list. Each list begins with the program type followed by the rest of the parameters used at your site.

Note: Special characters (anything other than numerics, alphabetic characters, $, #, or @) must be enclosed in quotation marks.

BTS Load Libraries Screen

The BTS Load Libraries screen is used to enter the dataset names of your BTS authorized load libraries. Do not enter dsnames for your application programs that are compiled to run under BTS. The BTS load libraries are system datasets that contain all the load modules that make up the BTS software.



```
Profile: DEFAULT ------ XPEDITER/TSO - BTS LOAD LIBRARIES ---------------------
COMMAND ===>

BTS (Program Product) Load Library DSNAMEs:

      (1) ===> 'IMS.V130X.BTSLIB'
      (2) ===>
      (3) ===>
      (4) ===>
      (5) ===>
      (6) ===>




  Note: Changes made to this screen override installed defaults


           Press ENTER to Process  or  Enter END Command to Terminate
```

Figure C-27. BTS Load Libraries Screen

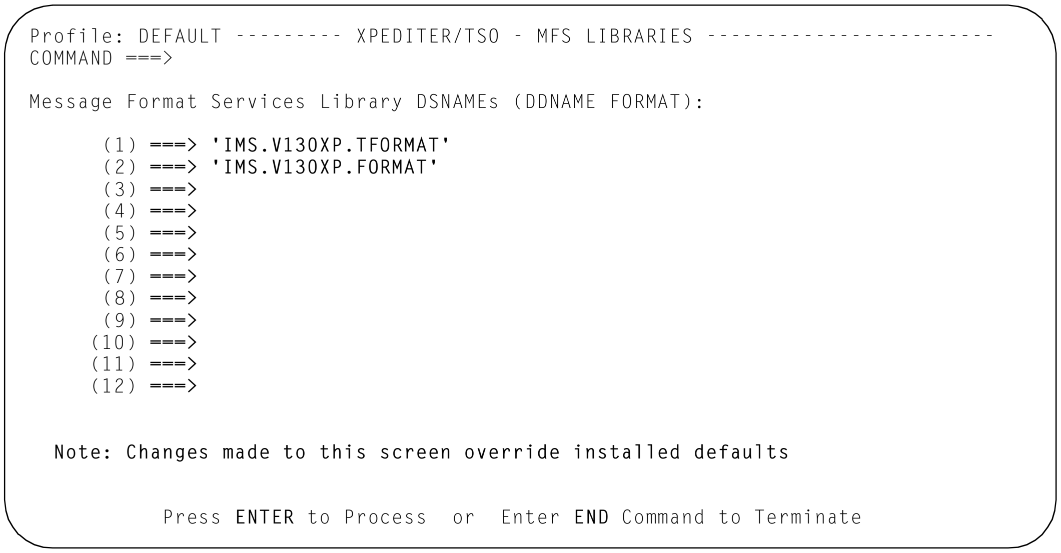
The field on this screen is:

BTS (Program Product) Load Library DSNAMEs

Enter from one to six dsnames. Consult a batch BTS PROC at your site if you are not sure which dsnames to enter.

MFS Libraries (Format) Screen

The MFS Libraries screen is used to enter your format library dataset names. The message format service helps to format messages that are transmitted to and from display screens. These message definitions are stored in the format library.



```
Profile: DEFAULT --------- XPEDITER/TSO - MFS LIBRARIES -----------------------
COMMAND ===>

Message Format Services Library DSNAMEs (DDNAME FORMAT):

      (1) ===> 'IMS.V130XP.TFORMAT'
      (2) ===> 'IMS.V130XP.FORMAT'
      (3) ===>
      (4) ===>
      (5) ===>
      (6) ===>
      (7) ===>
      (8) ===>
      (9) ===>
     (10) ===>
     (11) ===>
     (12) ===>


  Note: Changes made to this screen override installed defaults


           Press ENTER to Process  or  Enter END Command to Terminate
```

Figure C-28. MFS Libraries Screen

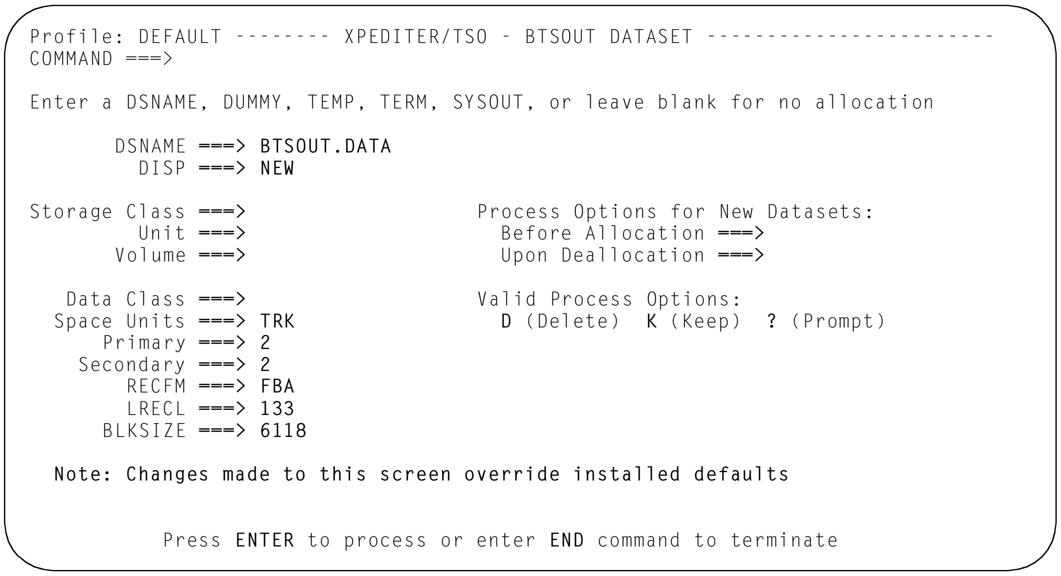
The field on this screen is:

Message Format Services Library DSNAMEs

Enter from one to twelve dsnames. Consult a batch BTS PROC at your site if you are unsure which dsnames to enter. These DSNAME fields can be left blank if the application to be debugged does not use MFS.

BTS Output (BTSOUT Dataset) Screen

The BTSOUT Dataset screen is used to allocate a sequential dataset to the BTSOUT ddname. The BTS input and output screen images and output messages are written to this BTS output listing. You can retain a copy of the BTSOUT dataset or, if it was allocated with SYSOUT=A, you can print a hard copy.



```
Profile: DEFAULT -------- XPEDITER/TSO - BTSOUT DATASET -----------------------
COMMAND ===>

Enter a DSNAME, DUMMY, TEMP, TERM, SYSOUT, or leave blank for no allocation

       DSNAME ===> BTSOUT.DATA
         DISP ===> NEW

Storage Class ===>                    Process Options for New Datasets:
         Unit ===>                      Before Allocation ===>
       Volume ===>                      Upon Deallocation ===>

   Data Class ===>                    Valid Process Options:
  Space Units ===> TRK                  D (Delete)  K (Keep)  ? (Prompt)
      Primary ===> 2
    Secondary ===> 2
        RECFM ===> FBA
        LRECL ===> 133
      BLKSIZE ===> 6118

  Note: Changes made to this screen override installed defaults


           Press ENTER to process or enter END command to terminate
```

Figure C-29. BTSOUT Dataset Screen

The fields on this screen are:

DSNAME

Enter a qualified dsname (enclosed in quotes), DUMMY, TEMP, TERM, or SYSOUT (JES dataset), or leave this field blank. If the dsname is entered without quotes, the user ID (or other dsname high-level qualifier) is prefixed and quotes are added. If the DSNAME field is left blank, the dataset is not allocated.

Consult a batch BTS PROC at your site if you are not sure which dsname to enter. If the BTSOUT ddname is not to be allocated, leave the DSNAME field blank.

DISP

Valid values for the DISP field are:

S	Shared (SHR)
M	Modified (MOD)
O	Old
N	New

If you are allocating an existing dataset, you can specify a disposition of SHR, OLD, or MOD. If you do not enter a value in the DISP field, a SHR disposition is assumed. Datasets with a disposition of SHR can be concatenated.

If the DSNAME field contains a value of DUMMY, TEMP, TERM, or SYSOUT, leave the DISP field blank.

Enter a disposition of NEW for a new or temporary dataset. If you are allocating a new (DSNAME='qualified-dsname') or temporary file (DSNAME=TEMP), you must enter values for all the fields below except Volume and Unit, which are optional for a new or temporary file.

If you specify a new dataset, it is allocated before your debugging session begins and is deallocated immediately after your debugging session ends. The fields for the Process Options for New Datasets let you specify how the new dataset is to be allocated and deallocated.

For more information about NEW, OLD, MOD, or SHR dataset dispositions, refer to "Types of Files That Can Be Allocated" on page C-16.

Storage Class

Enter the storage class name defined by your site that contains the dataset attributes related to the storage occupied by the dataset. A storage class is displayed only when SMS (Storage Management Subsystem) is being used.

Unit
If your site requires it, you must supply a value for the Unit field. Otherwise, leave blank for the default value.

Volume
If your site requires it, you must supply a value for the Volume field. Also enter the volume if you are specifying an uncataloged dataset. Otherwise, leave blank for the default value.

Data Class
Enter the data class name defined by your site that contains the dataset attributes related to the allocation of the dataset. A data class is displayed only when SMS (Storage Management Subsystem) is being used.

Space Units
Valid values are TRK, CYL, or a block size (1-32760).

Primary
Valid values are 0-32760.

Secondary
Valid values are 0-32760.

RECFM
Valid values are FBM, FBA, FB, F, VBM, VBA, VBS, VS, VB, V, and U.

LRECL
Valid values are 0-32760 (omit if RECFM is U for undefined length records).

BLKSIZE
Valid values are 0-32760.

If you are allocating a DUMMY or SYSOUT file (DSNAME=DUMMY or DSNAME=SYSOUT), you must enter the block size.

Before Allocation
Valid process options are:

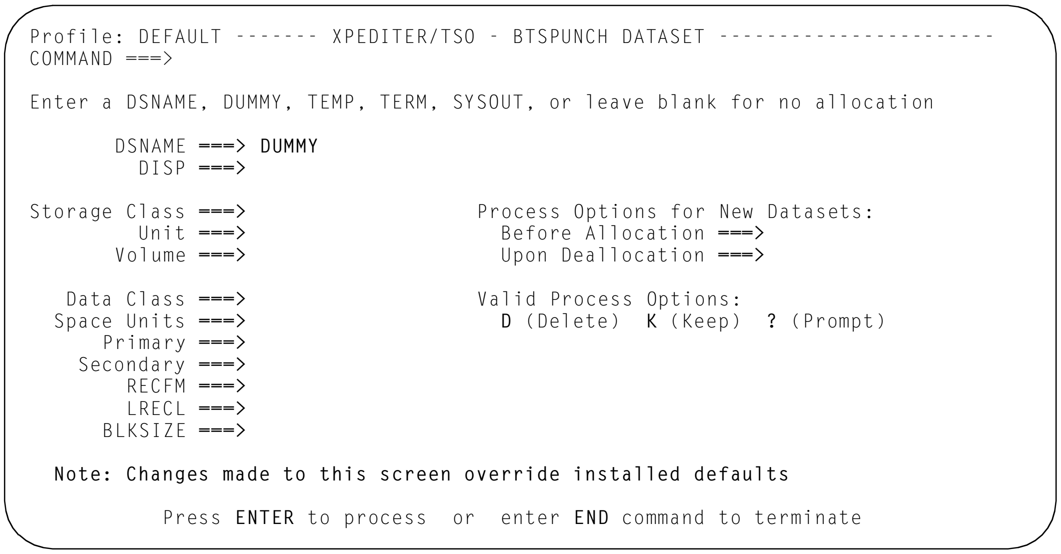
D	Automatically delete the old file, if one exists, before a new dataset is allocated.
K	Always keep the old dataset.
?	Display the Data Set Disposition screen each time a dataset is to be allocated, so that you can decide whether to delete or keep the old dataset.

Upon Deallocation
Valid process options are:

D	Automatically delete the new file after every debugging session.
K	Always keep the new dataset.
?	Display the Data Set Disposition screen each time a dataset is to be deallocated, so that you can decide whether to delete or keep the new dataset.

BTS Punch Output (BTSPUNCH Dataset) Screen

The BTSPUNCH Dataset screen is used to allocate a sequential dataset to the BTSPUNCH ddname. If executing as a TSO task, BTS attempts to open a sequential output dataset named BTSPUNCH. This dataset is used to create regression debugging input data. It contains everything that BTS receives as input.



```
Profile: DEFAULT ------- XPEDITER/TSO - BTSPUNCH DATASET ----------------------
COMMAND ===>

Enter a DSNAME, DUMMY, TEMP, TERM, SYSOUT, or leave blank for no allocation

       DSNAME ===> DUMMY
         DISP ===>

Storage Class ===>                   Process Options for New Datasets:
         Unit ===>                     Before Allocation ===>
       Volume ===>                     Upon Deallocation ===>

   Data Class ===>                   Valid Process Options:
  Space Units ===>                     D (Delete)  K (Keep)  ? (Prompt)
      Primary ===>
    Secondary ===>
        RECFM ===>
        LRECL ===>
      BLKSIZE ===>

  Note: Changes made to this screen override installed defaults

            Press ENTER to process  or  enter END command to terminate
```

Figure C-30. BTSPUNCH Dataset Screen

The fields on this screen are:

DSNAME

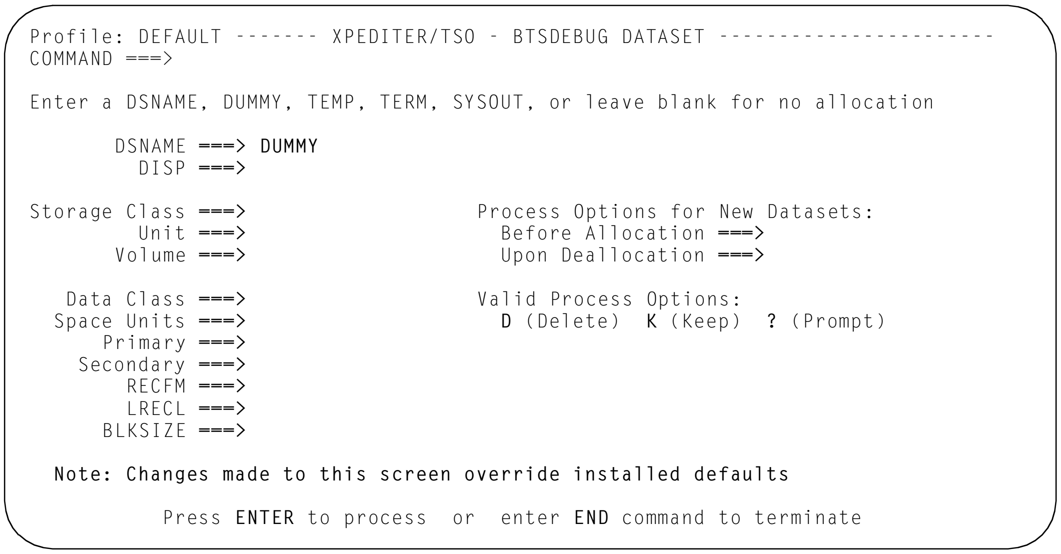
Enter a qualified dsname (enclosed in quotes), DUMMY, TEMP, TERM, or SYSOUT (JES dataset), or leave this field blank. If the dsname is entered without quotes, the user ID (or other dsname high-level qualifier) is prefixed and quotes are added. If the DSNAME field is left blank, the dataset is not allocated.

Consult a batch BTS PROC at your site if you are not sure which dsname to enter. If you decide not to use this dataset, leave the DSNAME field blank. Most sites allocate this ddname to DUMMY for debugging rather than leaving it blank. This conforms to common batch and debugging message region usage.

The valid values for the remaining fields on this screen are the same as those on the BTSOUT Dataset screen. Refer to "BTSOutput (BTSOUT Dataset) Screen" on page C-27 for descriptions of the fields.

BTS Debug (BTSDEBUG Dataset) Screen

The BTS Debug function is activated by allocating a dataset to the BTSDEBUG ddname. The BTSDEBUG dataset contains SNAP dumps of the Trace Table and various control blocks taken at critical points during BTS execution.



```
Profile: DEFAULT ------- XPEDITER/TSO - BTSDEBUG DATASET ----------------------
COMMAND ===>

Enter a DSNAME, DUMMY, TEMP, TERM, SYSOUT, or leave blank for no allocation

       DSNAME ===> DUMMY
         DISP ===>

Storage Class ===>                    Process Options for New Datasets:
         Unit ===>                      Before Allocation ===>
       Volume ===>                      Upon Deallocation ===>

   Data Class ===>                    Valid Process Options:
  Space Units ===>                      D (Delete)  K (Keep)  ? (Prompt)
      Primary ===>
    Secondary ===>
        RECFM ===>
        LRECL ===>
      BLKSIZE ===>

  Note: Changes made to this screen override installed defaults

           Press ENTER to process  or  enter END command to terminate
```

Figure C-31. BTSDEBUG Dataset Screen

The fields on this screen are:

DSNAME

Enter a qualified dsname (enclosed in quotes), TEMP, TERM, or SYSOUT (JES dataset), or leave this field blank. If the dsname is entered without quotes, the user ID (or other dsname high-level qualifier) is prefixed and quotes are added. If the DSNAME field is left blank, the dataset is not allocated.

Note: For performance reasons, IBM strongly recommends that you do **not** allocate BTSDEBUG to DUMMY.

DISP

Valid values for the DISP field are:

N New

O Old

M Modified (MOD)

S Shared (SHR)

If TERM or SYSOUT was entered in the DSNAME field, you do not have to supply a value for the DISP field. If you are allocating a SYSOUT file (DSNAME=SYSOUT), enter the block size in the BLKSIZE field.

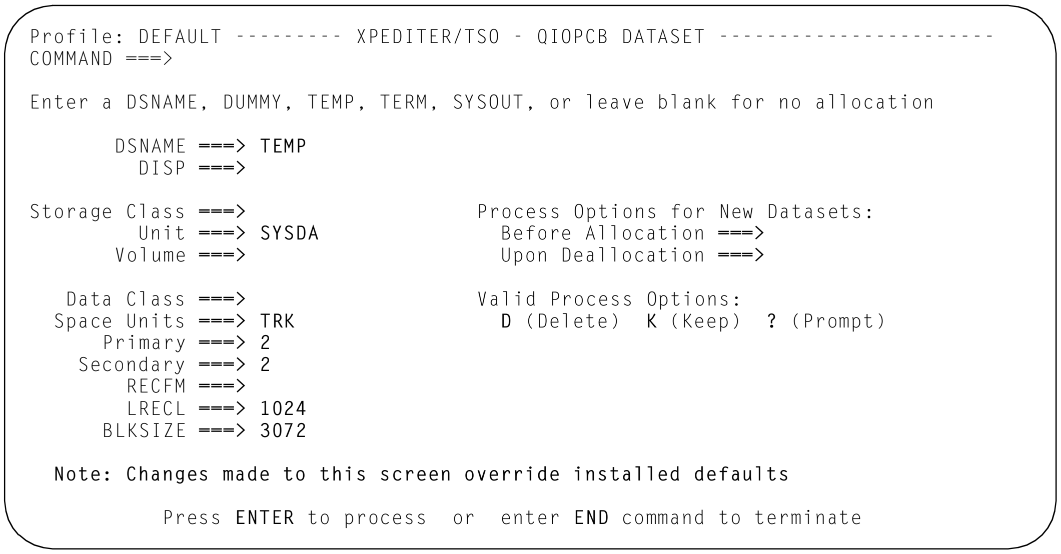
Datasets with a disposition of SHR can be concatenated.

Enter a disposition of NEW for a new or temporary dataset. If you are allocating a new (DSNAME='qualified-dsname') or temporary file (DSNAME=TEMP), you must enter values for all the remaining fields except Volume and Unit, which are optional for a new or temporary file. If you specify a new dataset, it is allocated before your debugging session begins and is deallocated immediately after your debugging session ends. The fields for the Process Options for New Datasets let you specify how the new dataset is to be allocated and deallocated.

The valid values for the remaining fields on this screen are the same as those on the BTSOUT Dataset screen. Refer to "BTS Output (BTSOUT Dataset) Screen" on page C-27 for descriptions of the fields.

BTS Work File (QIOPCB Dataset) Screen

The BTS QIOPCB Dataset screen is used to allocate a dataset to the QIOPCB ddname. The QIOPCB dataset is used for INSERT calls against IOPCBs.



```
Profile: DEFAULT --------- XPEDITER/TSO - QIOPCB DATASET ----------------------
COMMAND ===>

Enter a DSNAME, DUMMY, TEMP, TERM, SYSOUT, or leave blank for no allocation

       DSNAME ===> TEMP
         DISP ===>

Storage Class ===>                    Process Options for New Datasets:
         Unit ===> SYSDA                Before Allocation ===>
       Volume ===>                      Upon Deallocation ===>

   Data Class ===>                    Valid Process Options:
  Space Units ===> TRK                  D (Delete)  K (Keep)  ? (Prompt)
      Primary ===> 2
    Secondary ===> 2
        RECFM ===>
        LRECL ===> 1024
      BLKSIZE ===> 3072

  Note: Changes made to this screen override installed defaults

           Press ENTER to process  or  enter END command to terminate
```

Figure C-32. QIOPCB Dataset Screen

The fields on this screen are:

DSNAME

Enter a qualified dsname (enclosed in quotes), DUMMY, TEMP, TERM, or SYSOUT (JES dataset), or leave this field blank. If the dsname is entered without quotes, the user ID (or other dsname high-level qualifier) is prefixed and quotes are added. If the DSNAME field is left blank, the dataset is not allocated.

Consult a batch BTS PROC at your site if you are not sure which dsname to enter. If the QIOPCB ddname is not required, you can leave the DSNAME field blank.

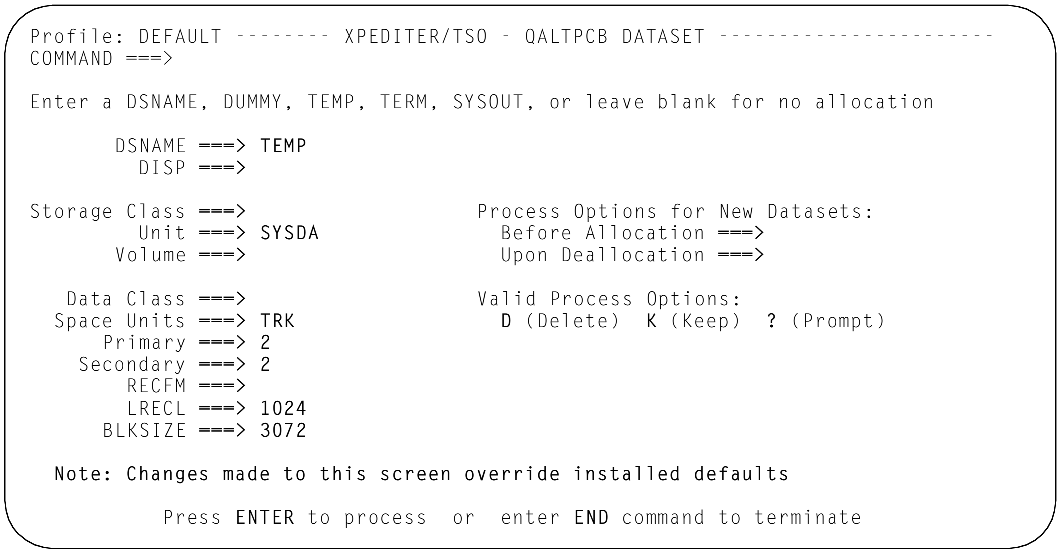
Most sites allocate this ddname to TEMP for debugging. Therefore, since this is a NEW dataset, the DCB attributes must be entered. The LRECL and BLKSIZE should match the largest used by any application.

Note: The prefilled defaults for LRECL and BLKSIZE are rarely large enough for actual application databases and transactions. Therefore, carefully check your output (and application usage) before specifying them.

The valid values for the remaining fields on this screen are the same as those on the BTSOUT Dataset screen. Refer to "BTS Output (BTSOUT Dataset) Screen on page C-27 for descriptions of the fields.

BTS Work File (QALTPCB Dataset) Screen

The QALTPCB Dataset screen is used to allocate a dataset to the QALTPCB ddname. The QALTPCB dataset is used for GET-UNIQUE or INSERT calls against alternate PCBs.



```
Profile: DEFAULT -------- XPEDITER/TSO - QALTPCB DATASET ----------------------
COMMAND ===>

Enter a DSNAME, DUMMY, TEMP, TERM, SYSOUT, or leave blank for no allocation

       DSNAME ===> TEMP
         DISP ===>

Storage Class ===>                   Process Options for New Datasets:
         Unit ===> SYSDA               Before Allocation ===>
       Volume ===>                     Upon Deallocation ===>

    Data Class ===>                  Valid Process Options:
   Space Units ===> TRK                D (Delete)  K (Keep)  ? (Prompt)
       Primary ===> 2
     Secondary ===> 2
         RECFM ===>
         LRECL ===> 1024
       BLKSIZE ===> 3072

  Note: Changes made to this screen override installed defaults

           Press ENTER to process  or  enter END command to terminate
```

Figure C-33. QALTPCB Dataset Screen

The fields on this screen are:

DSNAME

Enter a qualified dsname (enclosed in quotes), DUMMY, TEMP, TERM, or SYSOUT (JES dataset), or leave this field blank. If the dsname is entered without quotes, the user ID (or other dsname high-level qualifier) is prefixed and quotes are added. If the DSNAME field is left blank, the dataset is not allocated.

Consult a batch BTS PROC at your site if you are not sure which dsname to enter. If you do not plan to use the QALTPCB dataset, you can leave the DSNAME field blank.

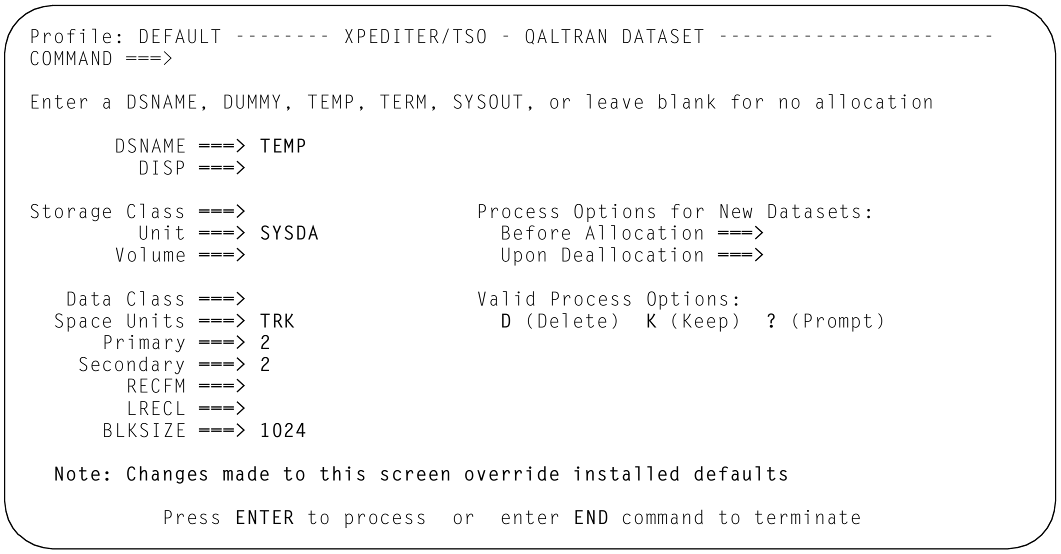
Most sites allocate this ddname to TEMP for debugging. Therefore, since this is a new dataset, the DCB attributes must be entered. The LRECL and BLKSIZE should match the largest used by any application.

Note: The prefilled defaults for LRECL and BLKSIZE are rarely large enough for actual application databases and transactions. Therefore, carefully check your output (and application usage) before specifying them.

The valid values for the remaining fields on this screen are the same as those on the BTSOUT Dataset screen. Refer to "BTS Output (BTSOUT Dataset) Screen" on page C-27 for descriptions of the fields.

BTS Work File (QALTRAN Dataset) Screen

The QALTRAN Dataset screen is used to allocate a dataset to the QALTRAN ddname. The QALTRAN dataset is a work file for alternate PCB output for a 3270 display screen or a 3270 printer.



```
Profile: DEFAULT -------- XPEDITER/TSO - QALTRAN DATASET -----------------------
COMMAND ===>

Enter a DSNAME, DUMMY, TEMP, TERM, SYSOUT, or leave blank for no allocation

       DSNAME ===> TEMP
         DISP ===>

Storage Class ===>                     Process Options for New Datasets:
         Unit ===> SYSDA                 Before Allocation ===>
       Volume ===>                       Upon Deallocation ===>

   Data Class ===>                     Valid Process Options:
  Space Units ===> TRK                   D (Delete)  K (Keep)  ? (Prompt)
      Primary ===> 2
    Secondary ===> 2
        RECFM ===>
        LRECL ===>
      BLKSIZE ===> 1024

  Note: Changes made to this screen override installed defaults

           Press ENTER to process  or  enter END command to terminate
```

Figure C-34. QALTRAN Dataset Screen

The fields on this screen are:

DSNAME

Enter a qualified dsname (enclosed in quotes), DUMMY, TEMP, TERM, or SYSOUT (JES dataset), or leave this field blank. If the dsname is entered without quotes, the user ID (or other dsname high-level qualifier) is prefixed and quotes are added. If the DSNAME field is left blank, the dataset is not allocated.

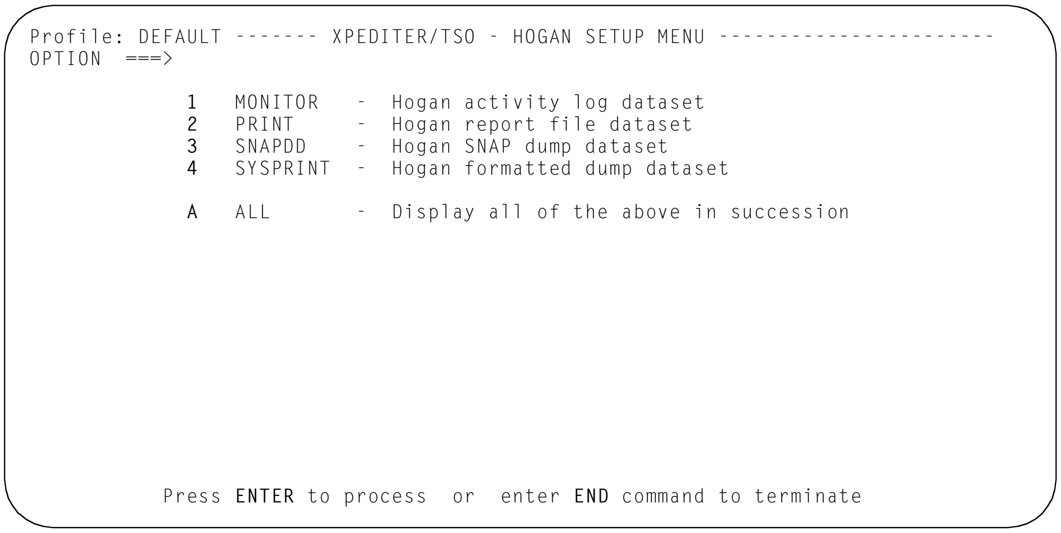
Consult a batch BTS PROC at your site if you are not sure which dsname to enter. If the QALTRAN ddname is not required, leave the DSNAME field blank.

Note: The prefilled defaults for LRECL and BLKSIZE are rarely large enough for actual application databases and transactions. Therefore, carefully check your output (and application usage) before specifying them.

The valid values for the remaining fields on this screen are the same as those on the BTSOUT Dataset screen. Refer to "BTS Output (BTSOUT Dataset) Screen" on page C-27 for descriptions of the fields.

Hogan Setup Menu

The menu that appears for option H (HOGAN) on the Hogan Setup Menu is shown in Figure C-35. The Hogan Setup Menu is used to select screens where you can list the datasets to be allocated to Hogan ddnames. The four screens can be selected individually or all together as a complete sequence. The screens for these options are described in the following subsections.



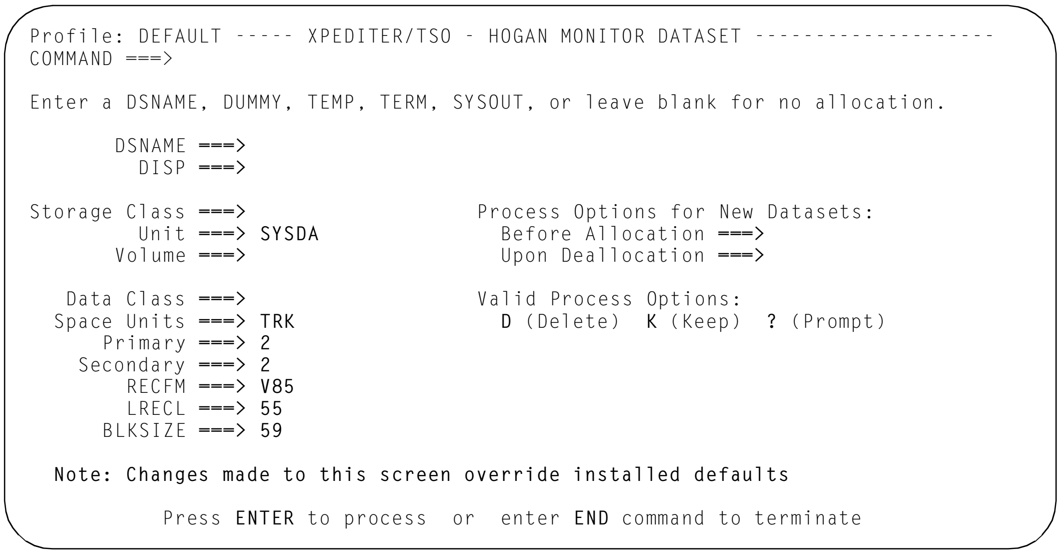
```
Profile: DEFAULT ------- XPEDITER/TSO - HOGAN SETUP MENU ----------------------
OPTION  ===>

             1   MONITOR   -  Hogan activity log dataset
             2   PRINT     -  Hogan report file dataset
             3   SNAPDD    -  Hogan SNAP dump dataset
             4   SYSPRINT  -  Hogan formatted dump dataset

             A   ALL       -  Display all of the above in succession









           Press ENTER to process  or  enter END command to terminate
```

Figure C-35. Hogan Setup Menu

Hogan Activity Log (Monitor Dataset) Screen

Data related to PEM calls is recorded in the Hogan monitor file. This file can be used as input to Hogan report programs that list, summarize, or analyze PEM activities. If it is not required, leave the DSNAME field blank.

If required, the Hogan monitor file is usually allocated as a variable block sequential file (VB). If you are using this file for debugging and expect an abend, an LRECL of 55 and a BLKSIZE of 59 are recommended to limit the loss of data in buffers. If an abend is not anticipated, then a larger block size is more efficient.



```
Profile: DEFAULT ----- XPEDITER/TSO - HOGAN MONITOR DATASET -------------------
COMMAND ===>

Enter a DSNAME, DUMMY, TEMP, TERM, SYSOUT, or leave blank for no allocation.

       DSNAME ===>
         DISP ===>

Storage Class ===>                 Process Options for New Datasets:
         Unit ===> SYSDA             Before Allocation ===>
       Volume ===>                   Upon Deallocation ===>

   Data Class ===>                 Valid Process Options:
  Space Units ===> TRK               D (Delete)  K (Keep)  ? (Prompt)
      Primary ===> 2
    Secondary ===> 2
        RECFM ===> V85
        LRECL ===> 55
      BLKSIZE ===> 59

  Note: Changes made to this screen override installed defaults

           Press ENTER to process  or  enter END command to terminate
```

Figure C-36. Hogan Monitor Dataset Screen

The fields on this screen are:

DSNAME

If it is not required, leave the DSNAME field blank.

If a dsname is required, enter a qualified dsname (enclosed in quotes), DUMMY, TEMP, TERM, or SYSOUT (JES dataset), or leave this field blank. If the dsname is entered without quotes, the user ID (or other dsname high-level qualifier) is prefixed and quotes are added. If the DSNAME field is left blank, the dataset is not allocated.

DISP

Valid values for the DISP field are:

S	Shared (SHR)
M	Modified (MOD)
O	Old
N	New

If you are allocating an existing dataset, you can specify a disposition of SHR, OLD, or MOD. If you do not enter a value in the DISP field, a SHR disposition is assumed. Datasets with a disposition of SHR can be concatenated.

If the DSNAME field contains a value of DUMMY, TEMP, TERM, or SYSOUT, leave the DISP field blank.

Enter a disposition of NEW for a new or temporary dataset. If you are allocating a new (DSNAME='qualified-dsname') or temporary file (DSNAME=TEMP), you must enter values for all the fields below except Volume and Unit, which are optional for a new or temporary file.

If you specify a new dataset, it is allocated before your debugging session begins and deallocated immediately after your debugging session ends. The fields for the Process Options for New Datasets let you specify how the new dataset is to be allocated and deallocated.

For more information about NEW, OLD, MOD, or SHR dataset dispositions, refer to "Types of Files That Can Be Allocated" on page A-16.

Storage Class

Enter the storage class name defined by your site that contains the dataset attributes related to the storage occupied by the dataset. A storage class is displayed only when SMS (Storage Management Subsystem) is being used.

Unit

If your site requires it, you must supply a value for the Unit field. Otherwise, leave blank for the default value.

Volume

If your site requires it, you must supply a value for the Volume field. Also enter the volume if you are specifying an uncataloged dataset. Otherwise, leave blank for the default value.

Data Class

Enter the data class name defined by your site that contains the dataset attributes related to the allocation of the dataset. A data class is displayed only when SMS (Storage Management Subsystem) is being used.

Space Units

Valid values are TRK, CYL, or a block size (1-32760).

Primary

Valid values are 0-32760.

Secondary

Valid values are 0-32760.

RECFM

If required, the Hogan monitor file is usually allocated as a variable block sequential file (VB).

LRECL

If you are using this file for debugging and expect an abend, an LRECL of 55 is recommended to limit the loss of data in buffers.

BLKSIZE
If you are using this file for debugging and expect an abend, a block size of 59 is recommended to limit the loss of data in buffers. If an abend is not anticipated, then a larger block size is more efficient.

If you are allocating a DUMMY or SYSOUT file (DSNAME=DUMMY or DSNAME=SYSOUT), you must enter the block size.

Before Allocation
Valid process options are:

D Automatically delete the old file, if one exists, before a new dataset is allocated.

K Always keep the old dataset.

? Display the Data Set Disposition screen each time a dataset is to be allocated, so that you can decide whether to delete or keep the old dataset.

Upon Deallocation
Valid process options are:

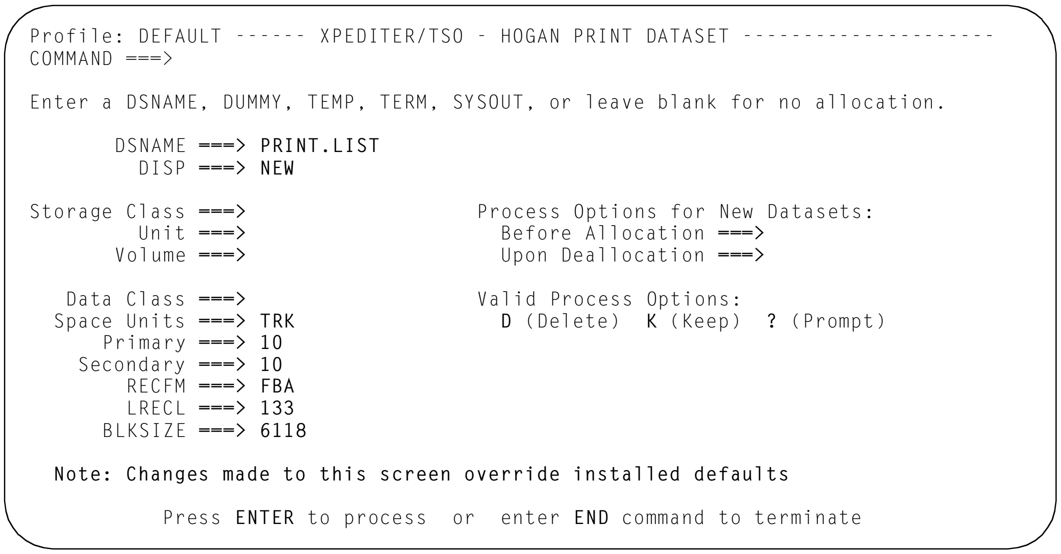
D Automatically delete the new file after every debugging session.

K Always keep the new dataset.

? Display the Data Set Disposition screen each time a dataset is to be deallocated, so that you can decide whether to delete or keep the new dataset.

Hogan Report File (Print Dataset) Screen

The Hogan print file is used frequently by Hogan programs to store reports or other output from application programs.



```
Profile: DEFAULT ------ XPEDITER/TSO - HOGAN PRINT DATASET --------------------
COMMAND ===>

Enter a DSNAME, DUMMY, TEMP, TERM, SYSOUT, or leave blank for no allocation.

       DSNAME ===> PRINT.LIST
         DISP ===> NEW

Storage Class ===>                  Process Options for New Datasets:
         Unit ===>                    Before Allocation ===>
       Volume ===>                    Upon Deallocation ===>

   Data Class ===>                  Valid Process Options:
  Space Units ===> TRK                D (Delete)  K (Keep)  ? (Prompt)
      Primary ===> 10
    Secondary ===> 10
        RECFM ===> FBA
        LRECL ===> 133
      BLKSIZE ===> 6118

  Note: Changes made to this screen override installed defaults

           Press ENTER to process  or  enter END command to terminate
```

Figure C-37. Hogan Print Dataset Screen

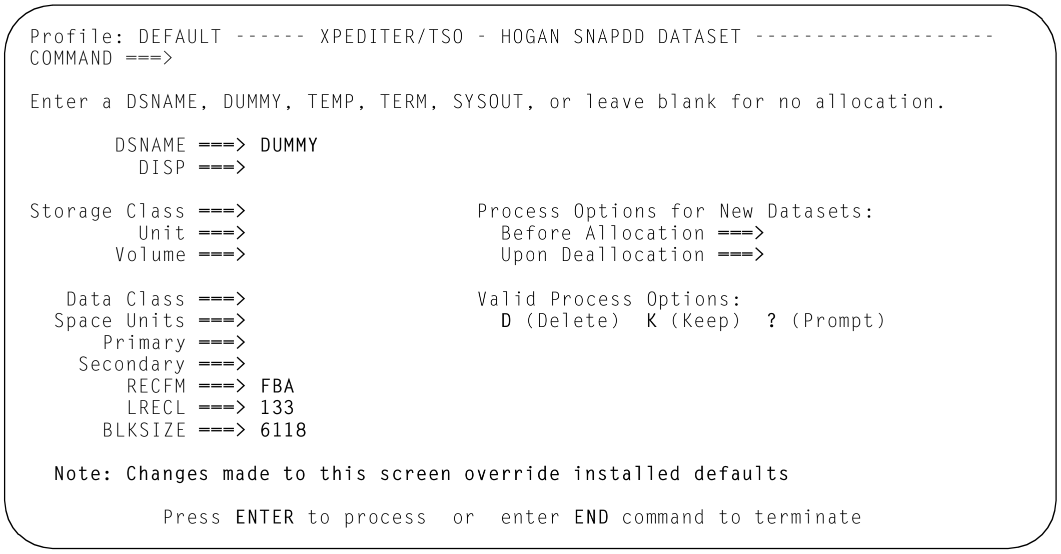
If needed, this Hogan file is usually allocated with a RECFM of FBA, LRECL of 133, and BLKSIZE of 6118.

Consult a batch Hogan PROC at your site if you are unsure of the dsname or other parameters. If the PRINT ddname is not needed, leave the DSNAME field blank.

The fields on this screen are similar to the Monitor screen. Refer to "Hogan Activity Log (Monitor Dataset) Screen" on page C-34 for a description of these fields.

Hogan SNAP Dump (SNAPDD Dataset) Screen

The Hogan SNAPDD file is used by Hogan to store SNAP dumps. Unlike a SYSUDUMP, a SNAP dump is not usually a printout of the entire region. A SNAP dump provides a "snap-shot" of the particular storage pool you want to see, as specified by your Hogan dump options.



```
Profile: DEFAULT ------ XPEDITER/TSO - HOGAN SNAPDD DATASET -------------------
COMMAND ===>

Enter a DSNAME, DUMMY, TEMP, TERM, SYSOUT, or leave blank for no allocation.

       DSNAME ===> DUMMY
         DISP ===>

Storage Class ===>                  Process Options for New Datasets:
         Unit ===>                    Before Allocation ===>
       Volume ===>                    Upon Deallocation ===>

   Data Class ===>                  Valid Process Options:
  Space Units ===>                    D (Delete)  K (Keep)  ? (Prompt)
      Primary ===>
    Secondary ===>
        RECFM ===> FBA
        LRECL ===> 133
      BLKSIZE ===> 6118

  Note: Changes made to this screen override installed defaults

           Press ENTER to process  or  enter END command to terminate
```

Figure C-38. Hogan SNAPDD Dataset Screen

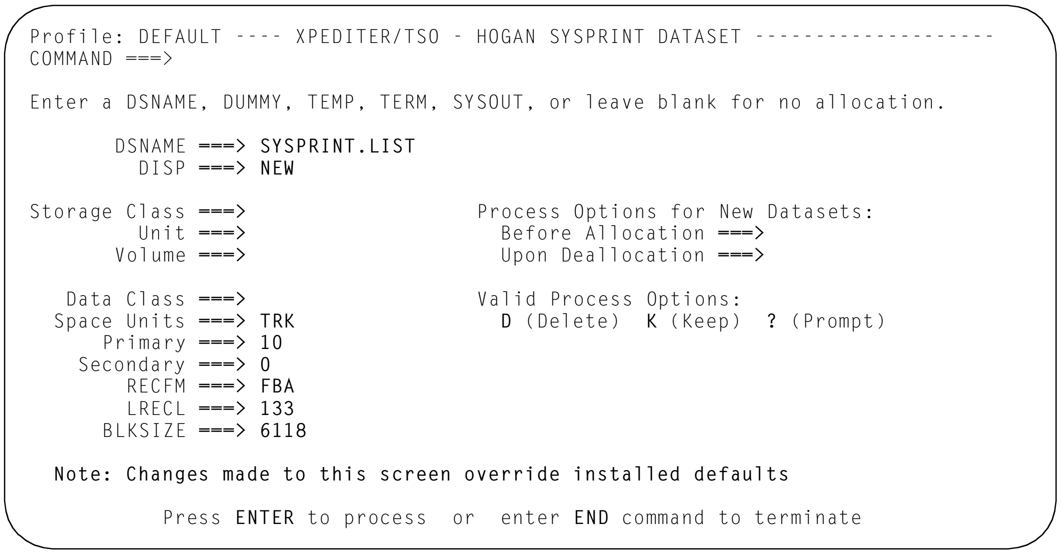
The recommended DCB parameters for the SNAPDD dataset are RECFM=FBA, BLKSIZE=6118, and LRECL=133.

Consult a batch Hogan PROC at your site if you are unsure of the dsname. If a SNAPDD file is not required, leave the DSNAME field blank.

The fields on this screen are similar to the Monitor screen. Refer to "Hogan Activity Log (Monitor Dataset) Screen" on page C-34 for a description of these fields.

Hogan Formatted Dump (SYSPRINT Dataset) Screen

The Hogan SYSPRINT file is used by Hogan to store PEM formatted dumps. Before running a transaction, set up your Hogan dump options to print the contents of certain control blocks or areas in main memory when an abend occurs.



```
Profile: DEFAULT ---- XPEDITER/TSO - HOGAN SYSPRINT DATASET -------------------
COMMAND ===>

Enter a DSNAME, DUMMY, TEMP, TERM, SYSOUT, or leave blank for no allocation.

       DSNAME ===> SYSPRINT.LIST
         DISP ===> NEW

Storage Class ===>                    Process Options for New Datasets:
         Unit ===>                      Before Allocation ===>
       Volume ===>                      Upon Deallocation ===>

   Data Class ===>                    Valid Process Options:
  Space Units ===> TRK                  D (Delete)  K (Keep)  ? (Prompt)
      Primary ===> 10
    Secondary ===> 0
        RECFM ===> FBA
        LRECL ===> 133
      BLKSIZE ===> 6118

  Note: Changes made to this screen override installed defaults

           Press ENTER to process  or  enter END command to terminate
```

Figure C-39. Hogan SYSPRINT Dataset Screen

The recommended DCB parameters for the SYSPRINT dataset are RECFM=FBA, BLKSIZE=6118, and LRECL=133.

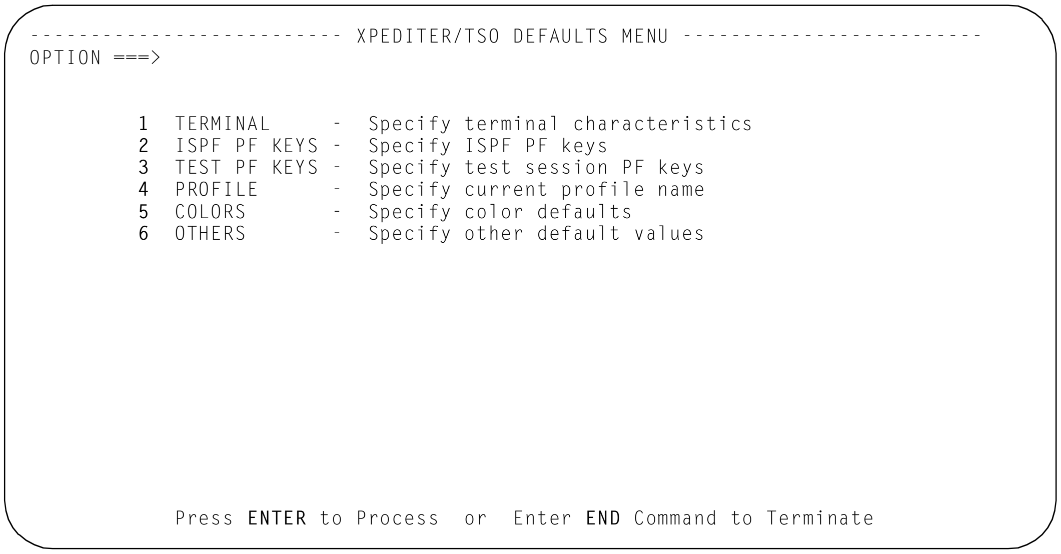
Consult a batch Hogan PROC at your site if you are unsure of the dsname. Most sites allocate this ddname to DUMMY for debugging rather than leaving it blank. This conforms to common batch and debugging message region usage.

The fields on this screen are similar to the Monitor screen. Refer to "Hogan Activity Log (Monitor Dataset) Screen on page C-34 for a description of these fields.

Appendix D. Specifying Session Defaults

You can set or override the defaults to be used during debugging sessions with option 0 (DEFAULTS) on the XPEDITER/TSO Primary Menu. The XPEDITER/TSO Defaults Menu shown in Figure D-1 is displayed. The defaults you specify remain in effect across all XPEDITER/TSO and XPEDITER/IMS debugging sessions.

The Defaults Menu



```
------------------------- XPEDITER/TSO DEFAULTS MENU ------------------------
OPTION ===>


         1  TERMINAL     -  Specify terminal characteristics
         2  ISPF PF KEYS -  Specify ISPF PF keys
         3  TEST PF KEYS -  Specify test session PF keys
         4  PROFILE      -  Specify current profile name
         5  COLORS       -  Specify color defaults
         6  OTHERS       -  Specify other default values














            Press ENTER to Process  or  Enter END Command to Terminate
```

Figure D-1. XPEDITER/TSO Defaults Menu

From this menu you can select to:

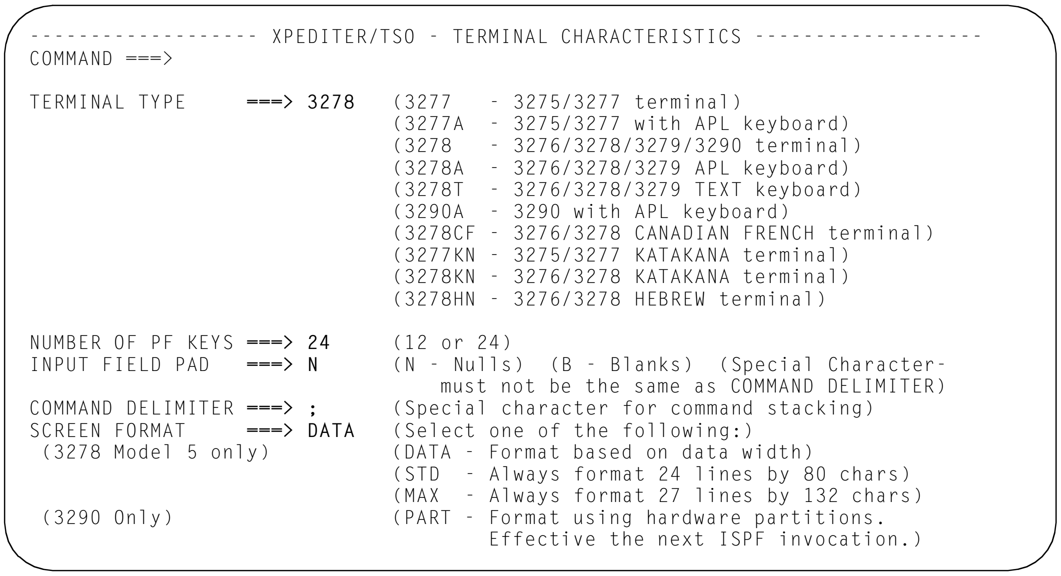
1. Specify the terminal type that is used in debugging sessions with XPEDITER/TSO.
2. Modify the ISPF PF key settings from the previous defaults. The terminal type and number of keys shown can be set with option 1.
3. Modify XPEDITER PF key settings used during your debugging session.
4. Select a profile to be used and tailor user profiles for all your different libraries and environments.
5. Specify color selections on a color terminal.
6. Specify session defaults, such as the default dsname high-level qualifier, and enable the jump function.

Specifying Terminal Characteristics

Select option **1** (TERMINAL) from the Defaults Menu to display the Terminal Characteristics screen shown in Figure D-2 on page D-2.

The Terminal Characteristics screen lets you specify the characteristics of the terminal you are using. You can specify the terminal type, number of PF keys, input field pad character, command delimiter, and screen format (for 3278 Model 5 or 3290 terminals). **It is recommended that you use the defaults.**

Changes that you make to these parameters take effect immediately and are saved in your user profile.



```
------------------- XPEDITER/TSO - TERMINAL CHARACTERISTICS -------------------
COMMAND ===>

TERMINAL TYPE     ===> 3278   (3277   - 3275/3277 terminal)
                              (3277A  - 3275/3277 with APL keyboard)
                              (3278   - 3276/3278/3279/3290 terminal)
                              (3278A  - 3276/3278/3279 APL keyboard)
                              (3278T  - 3276/3278/3279 TEXT keyboard)
                              (3290A  - 3290 with APL keyboard)
                              (3278CF - 3276/3278 CANADIAN FRENCH terminal)
                              (3277KN - 3275/3277 KATAKANA terminal)
                              (3278KN - 3276/3278 KATAKANA terminal)
                              (3278HN - 3276/3278 HEBREW terminal)

NUMBER OF PF KEYS ===> 24     (12 or 24)
INPUT FIELD PAD   ===> N      (N - Nulls)  (B - Blanks)  (Special Character-
                                  must not be the same as COMMAND DELIMITER)
COMMAND DELIMITER ===> ;      (Special character for command stacking)
SCREEN FORMAT     ===> DATA   (Select one of the following:)
 (3278 Model 5 only)          (DATA - Format based on data width)
                              (STD  - Always format 24 lines by 80 chars)
                              (MAX  - Always format 27 lines by 132 chars)
 (3290 Only)                  (PART - Format using hardware partitions.
                                      Effective the next ISPF invocation.)
```

Figure D-2. Terminal Characteristics Screen

The fields on this screen are:

TERMINAL TYPE

The valid terminal types that can be entered on the terminal characteristics panel are listed in Figure D-2. Some of these types may not apply to your installation and may not be displayed on the panel.

To assure correct operation of ISPF, the terminal type must be correct. The TERMINAL TYPE field is used to determine the character set for your terminal. ISPF automatically determines the terminal type during ISPF initialization and sets it to the appropriate value. The terminal type is displayed on the ISPF Primary Option Menu and the ISPF Parameter Options screen for reference. It is also displayed and can be modified on the PF Key Definition screen.

NUMBER OF PF KEYS

The number of PF keys is used to determine the set of PF key definitions to be used for your terminal. To assure correct operation of ISPF, this number must be correct. The number of PF keys is displayed on the ISPF Primary Option Menu and the ISPF Parameter Options screen for reference. It is also displayed and can be modified on the PF Key Definition screen.

Specify one of the following:

12 If you are using a display station with 12 PF keys or less.

24 If you are using a display station with 24 PF keys.

INPUT FIELD PAD

Specify one of the following:

N	If you want screen input fields to be padded with nulls.
B	If you want screen input fields to be padded with blanks.
character	If you want screen input fields to be padded with a special character. This special character **must** be different than the command delimiter character.

COMMAND DELIMITER

Select the special character you want to use as a command delimiter. This delimiter is used to separate stacked commands entered in the command or option field on a screen or entered by means of a PF key.

The character chosen for the command delimiter function must not be the same as the character chosen for input field padding. The special characters, equals (=) and period (.), are reserved for ISPF functions. Alphabetic and alphanumeric characters are not valid delimiters.

The following example shows the use of a semicolon as a command delimiter:

```
COMMAND ===> nulls; tabs /;unnum;tso listc;tso status
```

SCREEN FORMAT

For a *3278 Model 5 terminal*, specify one of the following:

DATA If you want the screen size to be based on the data width. If DATA is specified, ISPF automatically switches between "default" format (24 lines by 80 characters) and "native" format (27 lines by 132 characters), based on the width of the data to be displayed.

STD If you want the screen to be 80 characters wide. If STD is specified, ISPF holds the screen format constant.

MAX If you want the screen to be 132 characters wide. If MAX is specified, ISPF holds the screen format constant.

For a *3290 terminal*, specify **PART** if you want the screen size to be the maximum available. The effect of this parameter on a 3290 terminal is as follows:

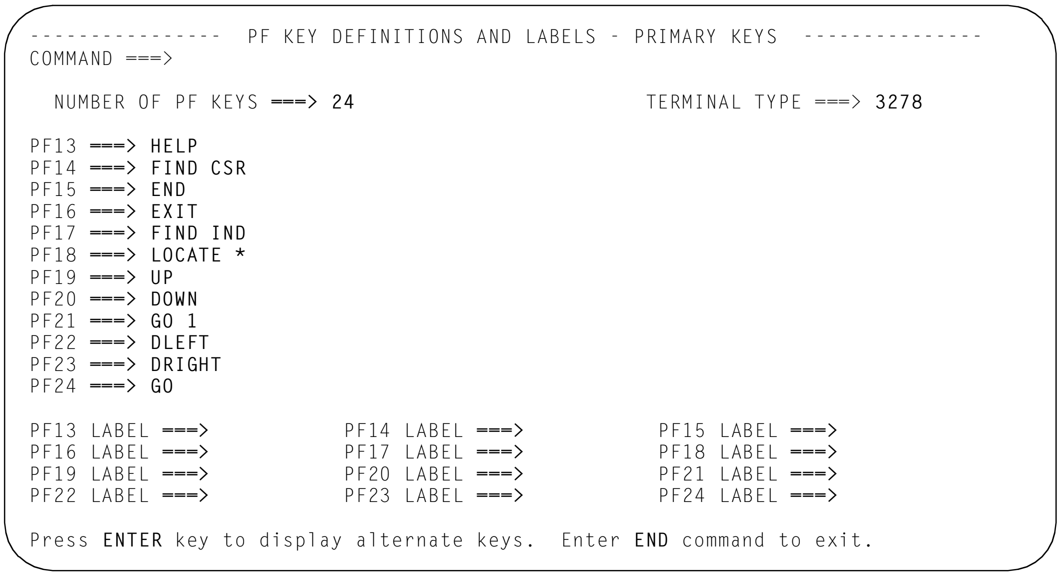
- If the 3290 is configured to support partitions, ISPF formats the screen as 31 rows by 80 columns, 62 rows by 80 columns, 31 rows by 160 columns, or 62 rows by 160 columns, depending on the screen configuration. In addition, the SPLITV command is enabled.
- If the 3290 is not configured to support hardware partitions, then the default mode depends on the definition of the terminal to the system.

Specifying PF Key Definitions

XPEDITER/TSO has two sets of PF keys. When using XPEDITER/TSO, you use ISPF PF key settings. These PF keys are used throughout the product, except when you are within a debugging session—then you use XPEDITER debugging session PF keys.

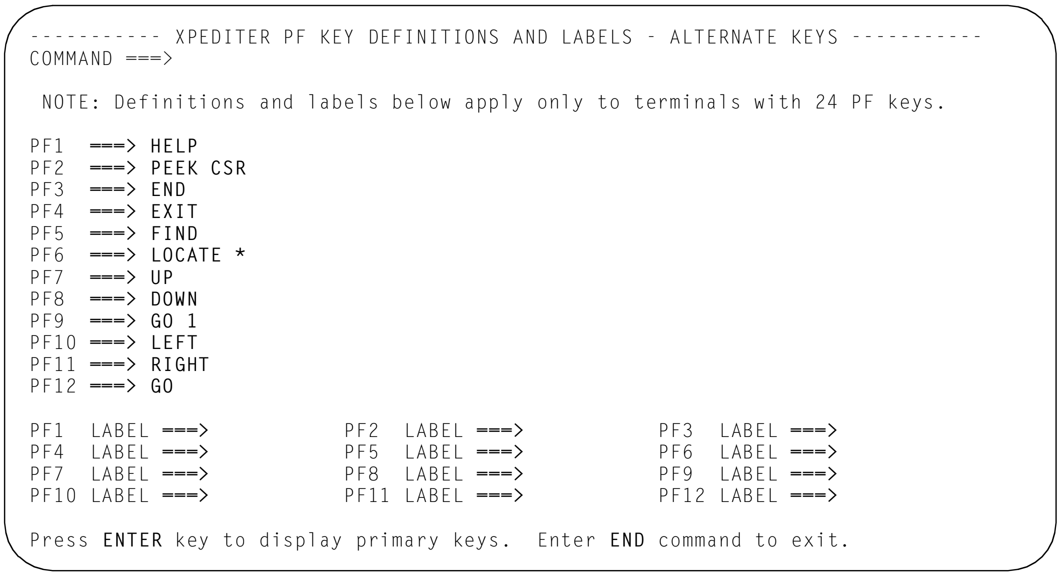
To modify the ISPF PF key settings, select option **2** (ISPF PF KEYS) from the Defaults Menu. The PF Key Definitions and Labels - Primary Keys screen shown in Figure D-3 on page D-4 is displayed. To modify the XPEDITER debugging session PF key settings, select option **3** (TEST PF KEYS). The XPEDITER PF Key Definitions and Labels - Alternate Keys screen shown in Figure Figure D-4 on page D-4 is displayed.

Note: Depending on the number of PF keys you specified on the Terminal Characteristics screen (Figure D-2 on page D-2), the appropriate set of PF keys is displayed on these screens.



```
---------------  PF KEY DEFINITIONS AND LABELS - PRIMARY KEYS  --------------
COMMAND ===>

  NUMBER OF PF KEYS ===> 24                        TERMINAL TYPE ===> 3278

PF13 ===> HELP
PF14 ===> FIND CSR
PF15 ===> END
PF16 ===> EXIT
PF17 ===> FIND IND
PF18 ===> LOCATE *
PF19 ===> UP
PF20 ===> DOWN
PF21 ===> GO 1
PF22 ===> DLEFT
PF23 ===> DRIGHT
PF24 ===> GO

PF13 LABEL ===>           PF14 LABEL ===>           PF15 LABEL ===>
PF16 LABEL ===>           PF17 LABEL ===>           PF18 LABEL ===>
PF19 LABEL ===>           PF20 LABEL ===>           PF21 LABEL ===>
PF22 LABEL ===>           PF23 LABEL ===>           PF24 LABEL ===>

Press ENTER key to display alternate keys.  Enter END command to exit.
```

Figure D-3. XPEDITER PF Key Definitions and Labels - Primary Keys Screen



```
----------- XPEDITER PF KEY DEFINITIONS AND LABELS - ALTERNATE KEYS -----------
COMMAND ===>

 NOTE: Definitions and labels below apply only to terminals with 24 PF keys.

PF1  ===> HELP
PF2  ===> PEEK CSR
PF3  ===> END
PF4  ===> EXIT
PF5  ===> FIND
PF6  ===> LOCATE *
PF7  ===> UP
PF8  ===> DOWN
PF9  ===> GO 1
PF10 ===> LEFT
PF11 ===> RIGHT
PF12 ===> GO

PF1  LABEL ===>           PF2  LABEL ===>           PF3  LABEL ===>
PF4  LABEL ===>           PF5  LABEL ===>           PF6  LABEL ===>
PF7  LABEL ===>           PF8  LABEL ===>           PF9  LABEL ===>
PF10 LABEL ===>           PF11 LABEL ===>           PF12 LABEL ===>

Press ENTER key to display primary keys.  Enter END command to exit.
```

Figure D-4. XPEDITER PF Key Definitions and Labels - Alternate Keys Screen

Changes can be made to the PF key designations by typing over the appropriate fields. PF keys can also be labeled, or given a descriptive name as well as the actual command mapped to the key.

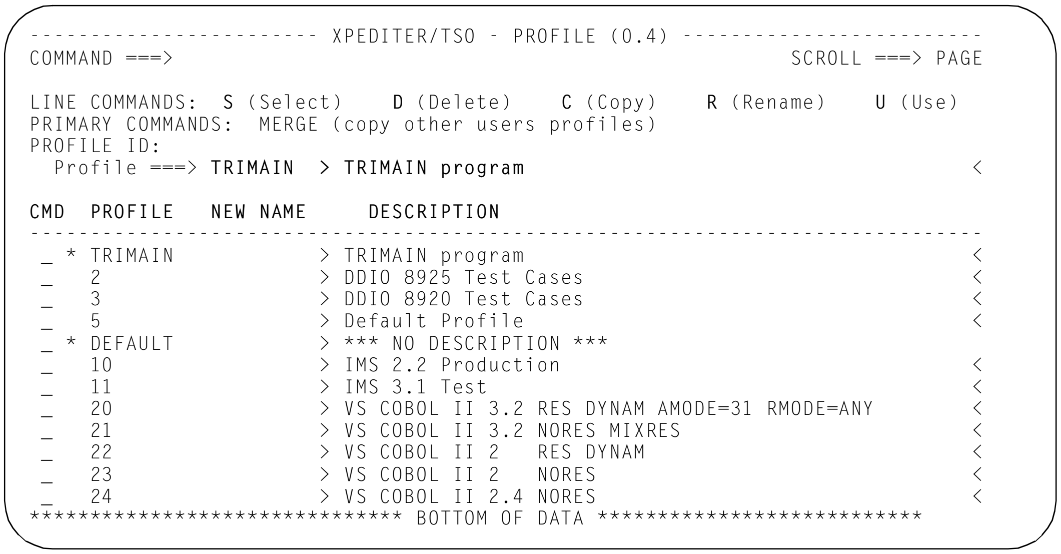
Specifying User Profiles

The user profile contains all the changes you make to the XPEDITER defaults. This includes installation defaults established by your system programming staff, and most importantly, environment parameters and setup options for your debugging session.

Select option **0** (Defaults) from the Primary Menu to display the Defaults Menu. Select option **4** (PROFILE) from the Defaults Menu to access the Profile screen shown in Figure D-5 on page D-5. This screen can also be reached by entering a question mark (?) as a profile identifier on the Primary Menu or by using the PROFILE command from a test screen. Refer to "Using the PROFILE Command" on page D-6 for additional information. This screen lets you:

- Switch to a new profile
- Change the description of a profile

- Make a particular profile current
- Delete a profile
- Copy a profile
- Rename a profile
- Use a profile



```
------------------------ XPEDITER/TSO - PROFILE (0.4) -------------------------
COMMAND ===>                                                    SCROLL ===> PAGE

LINE COMMANDS:  S (Select)    D (Delete)    C (Copy)    R (Rename)    U (Use)
PRIMARY COMMANDS:  MERGE (copy other users profiles)
PROFILE ID:
  Profile ===> TRIMAIN  > TRIMAIN program                                      <

CMD  PROFILE   NEW NAME     DESCRIPTION
-------------------------------------------------------------------------------
 _ * TRIMAIN            > TRIMAIN program                                      <
 _   2                  > DDIO 8925 Test Cases                                 <
 _   3                  > DDIO 8920 Test Cases                                 <
 _   5                  > Default Profile                                      <
 _ * DEFAULT            > *** NO DESCRIPTION ***
 _   10                 > IMS 2.2 Production                                   <
 _   11                 > IMS 3.1 Test                                         <
 _   20                 > VS COBOL II 3.2 RES DYNAM AMODE=31 RMODE=ANY         <
 _   21                 > VS COBOL II 3.2 NORES MIXRES                         <
 _   22                 > VS COBOL II 2   RES DYNAM                            <
 _   23                 > VS COBOL II 2   NORES                                <
 _   24                 > VS COBOL II 2.4 NORES                                <
***************************** BOTTOM OF DATA **************************
```

Figure D-5. XPEDITER Profile Screen

The Profile screen has a fixed (body) area and a scrollable area. The fixed body area has a command line and a PROFILE ID area that shows the current profile.

You can perform several operations in this area.

1. Use the SORT command on the command line to resort your profiles.
2. Use the PROFILE field to:

   Switch to a new profile: To do this, type over the profile identifier with a new identifier. The identifier can be any 8-character name or number. When you change the profile specified in this field, the profile change becomes effective immediately. This lets you execute debugging sessions with different environments, load libraries, levels of systems software, and so on.

   Change the description of a profile: To do this, type over the DESCRIPTION field. You can combine this with selecting a new profile, in which case, the description is associated with the new profile.

The scrollable area of the screen has a line for each existing profile and includes a line command area. An asterisk (*) preceding a profile identifier indicates the current profile. You can use the following line commands:

- **S** (Select)—to make a particular profile current. You can combine this command with a copy function by entering the identifier of the new profile in the NEW NAME field. This makes the new profile the current profile.
- **D** (Delete)—to delete a profile.
- **C** (Copy)—to copy a profile. Enter the identifier of the new profile in the NEW NAME field. Press **Enter**, and a new profile with the given identifier is created. If you also type over the DESCRIPTION field, the new description goes only with the new profile. If the new profile identifier already exists, you must verify that you really want to overwrite the profile.
- **R** (Rename)—to rename a profile. Enter the new profile identifier in the NEW NAME field. You can also change the description by typing over the DESCRIPTION field. Press **Enter**, and the profile is renamed.

- **U** (Use)—to use a profile. This line command takes you directly to XPEDITER/TSO with the selected profile.

Using the MERGE Command

The MERGE command copies profiles from an alternate profile dataset to the current profile dataset.

Using the PROFILE Command

The PROFILE command is available on the test screens to change the current profile without leaving the test screen itself. You can also use the PROFILE command from the Primary Menu. The syntax of the command is:

```
PROFile {xxxxxxxx}
        {?       }
```

Where *xxxxxxxx* is the profile identifier. Use this command as follows:

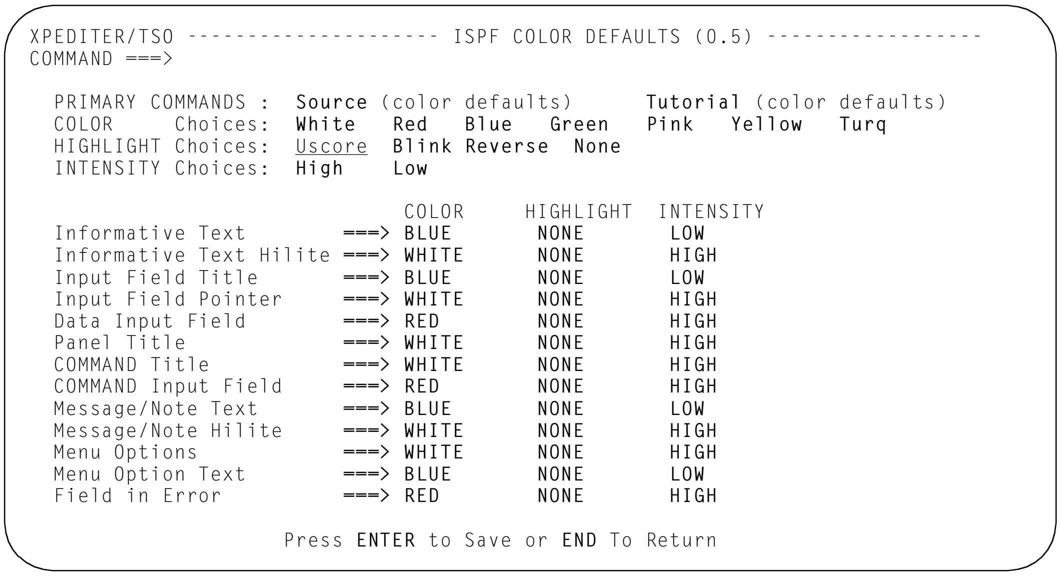
1. Enter the command and a profile identifier to change to another profile. If the profile identifier does not currently exist, the Profile screen shown in Figure D-5 on page D-5 is displayed so you can enter the description for the new profile or select a different profile identifier.
2. Enter the command without an identifier or with a question mark (?) to go to the Profile screen shown in Figure D-5 on page D-5.

When you leave the Profile screen, you are returned to the test screen, and the profile you selected is displayed in the message area. If you selected a new profile (i.e., a profile that did not currently exist), the Environments Menu is displayed.

Specifying Screen Colors

The next option on the Defaults Menu gives you color defaults for use with a color terminal. A full range of colors is offered for the various display areas on the screen, whether in ISPF or the Source display. Both intensity and highlight choices are available for each color. This option lets you change color, highlight, and intensity on all ISPF, source, and tutorial screens.

Select option **5** (COLORS) from the Defaults Menu to access the ISPF Color Defaults screen shown in Figure D-6 on page D-7. You can also enter the COLOR command anywhere within XPEDITER/TSO to invoke this screen.



```
XPEDITER/TSO -------------------- ISPF COLOR DEFAULTS (0.5) -----------------
COMMAND ===>

  PRIMARY COMMANDS :  Source (color defaults)      Tutorial (color defaults)
  COLOR     Choices:  White   Red   Blue   Green   Pink   Yellow   Turq
  HIGHLIGHT Choices:  Uscore  Blink Reverse  None
  INTENSITY Choices:  High    Low

                               COLOR     HIGHLIGHT  INTENSITY
  Informative Text        ===> BLUE       NONE       LOW
  Informative Text Hilite ===> WHITE      NONE       HIGH
  Input Field Title       ===> BLUE       NONE       LOW
  Input Field Pointer     ===> WHITE      NONE       HIGH
  Data Input Field        ===> RED        NONE       HIGH
  Panel Title             ===> WHITE      NONE       HIGH
  COMMAND Title           ===> WHITE      NONE       HIGH
  COMMAND Input Field     ===> RED        NONE       HIGH
  Message/Note Text       ===> BLUE       NONE       LOW
  Message/Note Hilite     ===> WHITE      NONE       HIGH
  Menu Options            ===> WHITE      NONE       HIGH
  Menu Option Text        ===> BLUE       NONE       LOW
  Field in Error          ===> RED        NONE       HIGH

                  Press ENTER to Save or END To Return
```

Figure D-6. ISPF Color Defaults Screen

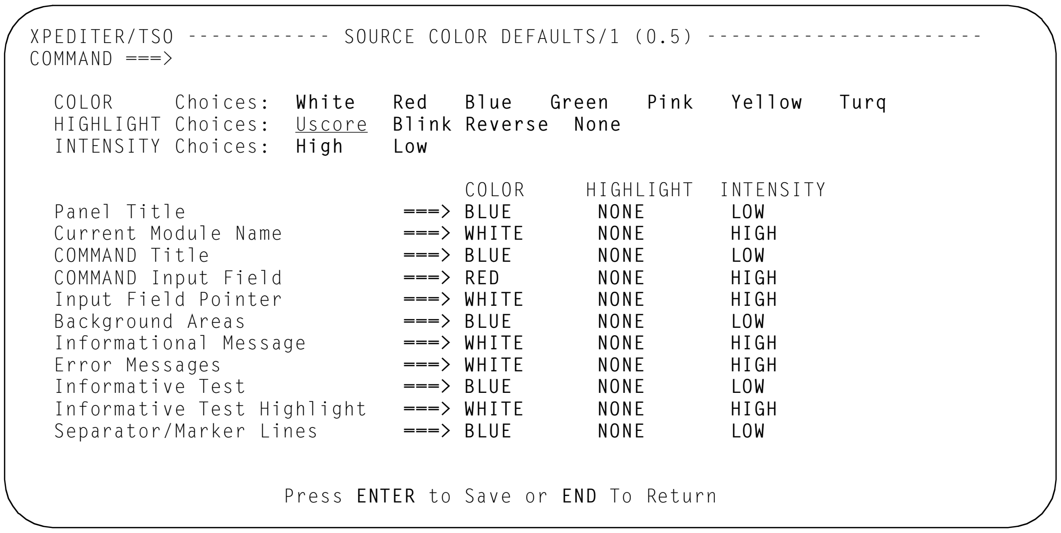
The entry values are listed on the screen. Note that color, highlight, and intensity choices can be indicated by the first character.

The following indicates some guidelines regarding the use of color:

- Colors take affect only on a 3279-B or ISPF-supported 7-color terminal.
- Enter the **ON** or **DEMO** command on the Color Default screens for an example of its use.
- Enter the **OFF** or **RESTORE** command to restore standard ISPF color settings.
- Intensity is ignored on IBM color terminals.
- Color is ignored on monochrome terminals.

Entering **S** or **SOURCE** from the ISPF Color Defaults screen invokes the Source Color Defaults/1 screen, shown in Figure D-7 on page D-8. You can also enter the COLOR S command anywhere within XPEDITER/TSO, except within a dialog debugging session, to invoke this screen.

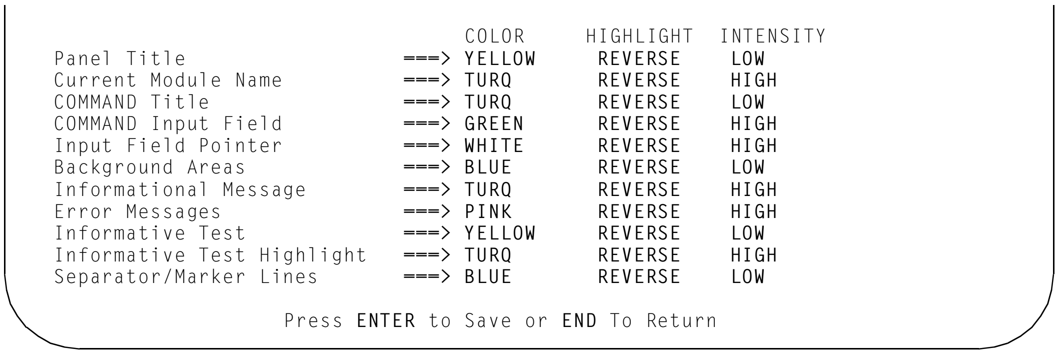
There are three screens for source color defaults. Press **Enter** without any changes to cycle through the screens.



```
XPEDITER/TSO ----------- SOURCE COLOR DEFAULTS/1 (0.5) ----------------------
COMMAND ===>

  COLOR     Choices:  White   Red   Blue   Green   Pink   Yellow   Turq
  HIGHLIGHT Choices:  Uscore  Blink Reverse  None
  INTENSITY Choices:  High    Low

                                  COLOR     HIGHLIGHT  INTENSITY
  Panel Title                ===> BLUE       NONE       LOW
  Current Module Name        ===> WHITE      NONE       HIGH
  COMMAND Title              ===> BLUE       NONE       LOW
  COMMAND Input Field        ===> RED        NONE       HIGH
  Input Field Pointer        ===> WHITE      NONE       HIGH
  Background Areas           ===> BLUE       NONE       LOW
  Informational Message      ===> WHITE      NONE       HIGH
  Error Messages             ===> WHITE      NONE       HIGH
  Informative Test           ===> BLUE       NONE       LOW
  Informative Test Highlight ===> WHITE      NONE       HIGH
  Separator/Marker Lines     ===> BLUE       NONE       LOW

                    Press ENTER to Save or END To Return
```

Figure D-7. Source Color Defaults/1 Screen

Typing **ON** or **DEMO** on the Source Color Defaults screen causes the COLOR, HIGH-LIGHT, and INTENSITY fields to automatically change to the following:

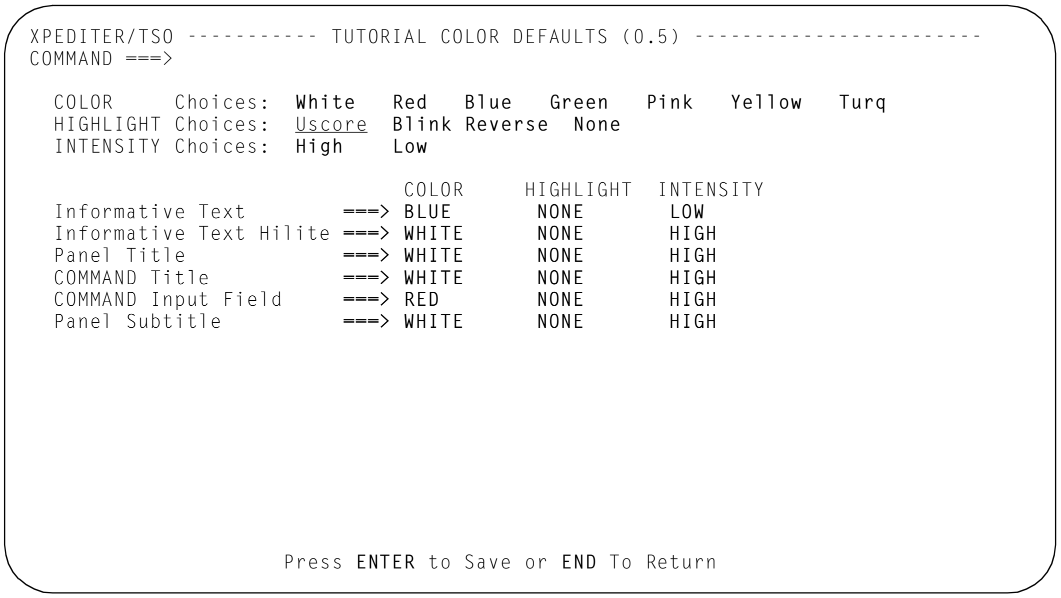


```
                                  COLOR     HIGHLIGHT  INTENSITY
  Panel Title                ===> YELLOW     REVERSE    LOW
  Current Module Name        ===> TURQ       REVERSE    HIGH
  COMMAND Title              ===> TURQ       REVERSE    LOW
  COMMAND Input Field        ===> GREEN      REVERSE    HIGH
  Input Field Pointer        ===> WHITE      REVERSE    HIGH
  Background Areas           ===> BLUE       REVERSE    LOW
  Informational Message      ===> TURQ       REVERSE    HIGH
  Error Messages             ===> PINK       REVERSE    HIGH
  Informative Test           ===> YELLOW     REVERSE    LOW
  Informative Test Highlight ===> TURQ       REVERSE    HIGH
  Separator/Marker Lines     ===> BLUE       REVERSE    LOW

                    Press ENTER to Save or END To Return
```

Figure D-8. Changing Source Color Defaults

Typing **OFF** or **RESTORE** causes the COLOR, HIGHLIGHT, and INTENSITY fields on the Source Color Defaults/1 screen to return to the default values shown in Figure D-7.

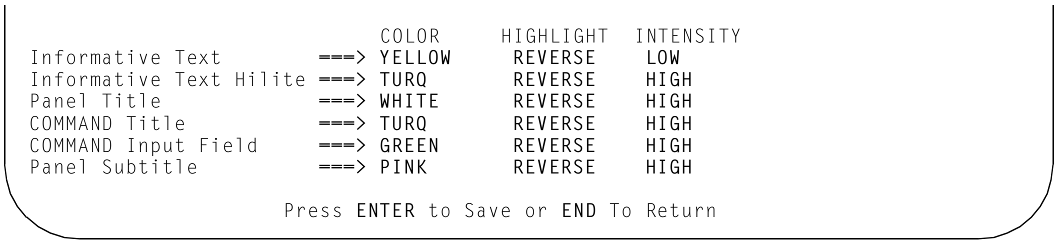
Entering **T** or **TUTORIAL** from the ISPF Color Defaults screen invokes the Tutorial Color Defaults screen, shown in Figure D-9 on page D-9. You can also enter the COLOR T command anywhere within XPEDITER/TSO to invoke this screen.



```
XPEDITER/TSO ---------- TUTORIAL COLOR DEFAULTS (0.5) -----------------------
COMMAND ===>

  COLOR     Choices:  White   Red   Blue   Green   Pink   Yellow   Turq
  HIGHLIGHT Choices:  Uscore  Blink Reverse  None
  INTENSITY Choices:  High    Low

                               COLOR     HIGHLIGHT  INTENSITY
  Informative Text        ===> BLUE       NONE       LOW
  Informative Text Hilite ===> WHITE      NONE       HIGH
  Panel Title             ===> WHITE      NONE       HIGH
  COMMAND Title           ===> WHITE      NONE       HIGH
  COMMAND Input Field     ===> RED        NONE       HIGH
  Panel Subtitle          ===> WHITE      NONE       HIGH










                     Press ENTER to Save or END To Return
```

Figure D-9. Tutorial Color Defaults Screen

Typing **ON** or **DEMO** in the Tutorial Color Defaults screen (Figure D-9) causes the COLOR, HIGHLIGHT, and INTENSITY fields to automatically change to the following:



```
                             COLOR     HIGHLIGHT  INTENSITY
Informative Text        ===> YELLOW     REVERSE    LOW
Informative Text Hilite ===> TURQ       REVERSE    HIGH
Panel Title             ===> WHITE      REVERSE    HIGH
COMMAND Title           ===> TURQ       REVERSE    HIGH
COMMAND Input Field     ===> GREEN      REVERSE    HIGH
Panel Subtitle          ===> PINK       REVERSE    HIGH

                     Press ENTER to Save or END To Return
```

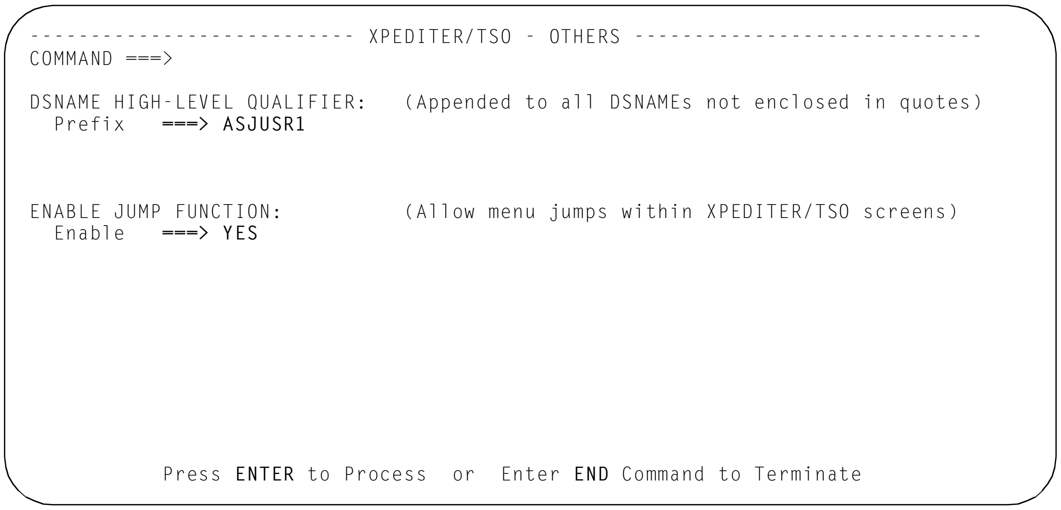
Figure D-10. Changing Tutorial Color Defaults

Typing **OFF** or **RESTORE** causes the COLOR, HIGHLIGHT, and INTENSITY fields on the Changing Tutorial Color Defaults screen (Figure D-10) to return to the default values shown in Figure D-9.

Specifying Other Default Values

Select option **6** (OTHERS) from the Defaults Menu to access the Specifying Other Default Values screen shown in Figure D-11 on page D-10. This screen does two things:

1. Sets up a default prefix for your dsname high-level qualifier that will be automatically appended to all dsnames not enclosed in quotes.
2. Enables the jump function.



```
--------------------------- XPEDITER/TSO - OTHERS -----------------------------
COMMAND ===>

DSNAME HIGH-LEVEL QUALIFIER:   (Appended to all DSNAMEs not enclosed in quotes)
  Prefix   ===> ASJUSR1



ENABLE JUMP FUNCTION:          (Allow menu jumps within XPEDITER/TSO screens)
  Enable   ===> YES








           Press ENTER to Process  or  Enter END Command to Terminate
```

Figure D-11. Specifying Other Default Values

The fields on this screen are as follows:

DSNAME HIGH-LEVEL QUALIFIER
The string entered here is automatically prefixed to any dsname not enclosed in quotes. Your user ID is the default value prefilled in this field.

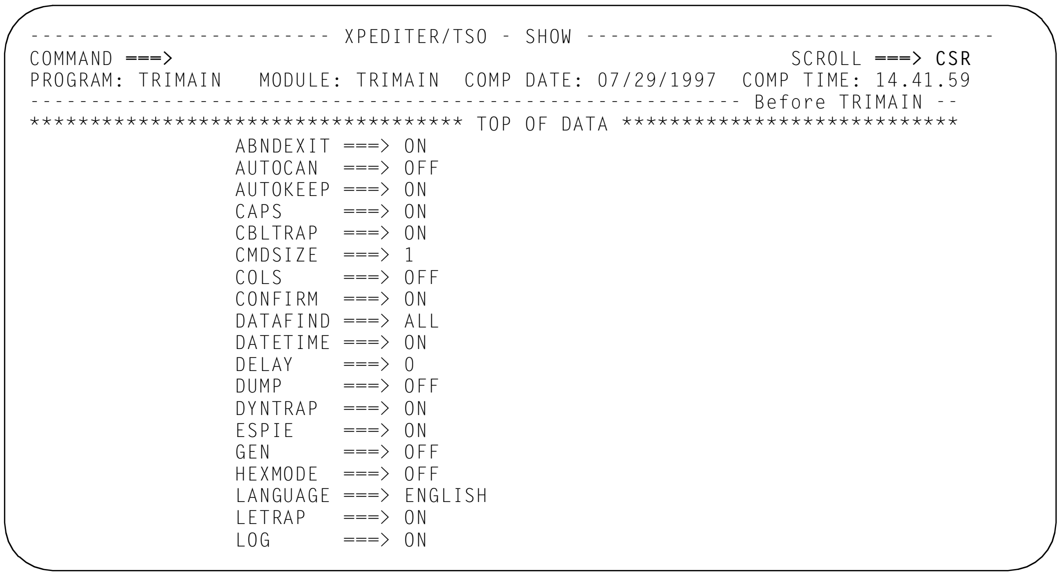
If your site requires a high-level qualifier that is *not* your user ID, enter that qualifier in the Prefix field. This will become your default high-level qualifier. Do not blank out this field. The prefix is used as the high-level qualifier in names generated by XPEDITER/TSO for the log and script datasets. Therefore, unless you explicitly name them, any attempt to allocate these datasets will fail. Also, in many sites, a dataset cannot be cataloged without a user ID.

ENABLE JUMP FUNCTION
Turning on the jump function (YES, the default) lets you move from one screen to another without passing through the Primary Menu. The jump function is the same as that provided inside ISPF.

The equal sign (=) is used to take you to a Primary Menu. Entering **=2** takes you to option 2 of the Primary Menu. If the jump function is turned on (enabled) and the equal sign convention is used, you are referred to the XPEDITER/TSO Primary Menu. If the jump function is off, you are referred to the ISPF Primary Menu.

Controlling Test Session Defaults (SET Commands)

You can use the SET commands to override a number of XPEDITER/TSO test session defaults in order to achieve your debugging objectives. For instance, you can control whether or not to intercept program abends, control the speed of program trace, or control the way XPEDITER/TSO deals with modules. The current status of the SET commands can be displayed by entering **SHOW SETS** or **SHOW OPTIONS** from the primary command line. Refer to Figure D-12 on page D-11. Some values are set only for the duration of the test session, while others are maintained across test sessions. All the SET commands that affect module management must be in effect before loading your program into memory. This can be done by editing an INCLUDE script that contains the SET commands and specifying it on the test screen as an initial script before entering the test session.



```
------------------------ XPEDITER/TSO - SHOW --------------------------------
COMMAND ===>                                                 SCROLL ===> CSR
PROGRAM: TRIMAIN   MODULE: TRIMAIN  COMP DATE: 07/29/1997  COMP TIME: 14.41.59
------------------------------------------------------------- Before TRIMAIN --
*********************************** TOP OF DATA ***************************
                ABNDEXIT ===> ON
                AUTOCAN  ===> OFF
                AUTOKEEP ===> ON
                CAPS     ===> ON
                CBLTRAP  ===> ON
                CMDSIZE  ===> 1
                COLS     ===> OFF
                CONFIRM  ===> ON
                DATAFIND ===> ALL
                DATETIME ===> ON
                DELAY    ===> 0
                DUMP     ===> OFF
                DYNTRAP  ===> ON
                ESPIE    ===> ON
                GEN      ===> OFF
                HEXMODE  ===> OFF
                LANGUAGE ===> ENGLISH
                LETRAP   ===> ON
                LOG      ===> ON
```

Figure D-12. SHOW SET/OPTIONS Screen

The SET command keywords are described below. The default option is underscored, and optional parameters are in parentheses.

- **SET ABENDSCR** *abend-script*

  The SET ABENDSCR command is valid in unattended batch. It contains the member name of the abend script; any name up to eight characters. The abend script must be a member of the PDS allocated to the ddname XINCLUDE. The script is executed when an abend occurs.

- **SET ABNDEXIT <u>ON</u>/OFF**

  Controls whether XPEDITER/TSO intercepts abends. If ABNDEXIT is set to ON, abends are intercepted and the user is given the opportunity to correct the error if the source for the failing module is available to XPEDITER. If ABNDEXIT is OFF, abends are not intercepted. Be aware that setting ABNDEXIT to OFF can result in unwanted dumps if any of the standard dump DD files have been allocated to the test session. The SET DUMP ON command is the recommended method for producing a dump.

  Note: If LETRAP is set to ON and the LE environment is active, ESPIE has no effect.

- **SET AUTOCAN ON/<u>OFF</u>**

  When AUTOCAN is set to ON, any dynamically called module not designated by the SET NOCANCEL command is deleted when control is returned from the module unless a breakpoint is set in it. A SET AUTOCAN command can be issued at any point within a test session. This command is valid in relation to modules compiled with OS/VS COBOL or CA-OPTIMIZER. It is not valid in VS COBOL II.

  SET AUTOCAN can be issued in both batch and interactive modes.

- **SET AUTOCLOS ON/<u>OFF</u>**

  With complex programs, an SC03 abend will occasionally occur when exiting an XPEDITER test. This abend is an indication that a file has not been closed and the system is having problems closing it. This may be prevented, in certain cases, by setting AUTOCLOS ON, prior to the exit from the test. This will cause XPEDITER to perform closes on any DCBs left open at the end of the test, as long as the DCBs are still available.

  CAUTION: If the SC03 abend is the result of an open DCB being freemained or overlaid, then the DCB will no longer be available, the file cannot be closed, and the SC03 abend is inevitable.

- **SET AUTOKEEP <u>ON</u>/OFF**

  Controls whether the data referenced in the current statement is automatically displayed in the Keep window. To configure the window size and placement of automatically kept items, refer to the SET (WINDOW) AUTOKEEP/KEEP/SOURCE *n* commands.

- **SET BRCOV ON/<u>OFF</u>**

  Controls Branch Coverage for a Code Coverage test. If Branch Coverage is desired by your site, set BRCOV=ON.

- **SET CAPS <u>ON</u>/OFF**

  The SET CAPS command handles user input conversion to uppercase when XPEDITER/TSO reads it from modifiable data fields such as Peek and Keep. XPEDITER/TSO defaults to converting all lowercase to uppercase before reading in user input; however, SET CAPS OFF disables the conversion and reads in user input as is. Use SET CAPS OFF in conjunction with SET LOWCASE ASIS if your terminal supports lowercase; otherwise, XPEDITER/TSO defaults to converting user modifiable fields to uppercase before writing to the terminal. The SET CAPS command is maintained across test sessions.

- **SET CBLTRAP <u>ON</u>/OFF**

  CBLTRAP should never be set to OFF except by XPEDITER/TSO technicians using it for diagnostic purposes. Changing the default from ON to OFF deactivates both the XPEDITER/TSO support that detects missing DD statements and also the support that opens windows at the bottom of the screen for files allocated to the terminal.

- **SET CMDSIZE <u>1</u>/2/3**

  SET CMDSIZE controls the size of the primary command area in case one line is not long enough and, once set, it is maintained across test sessions. Entering SET CMDSIZE 2 from the primary command line immediately expands the command line size to two lines. SET CMDSIZE 3 expands the command line size to three lines. When you are using more than a single command line, you can type commands continuously and XPEDITER/TSO wraps them around the end of the line. You can default the primary command area to other than a single line by editing an INCLUDE script to contain SET CMDSIZE *n* and specifying the INCLUDE member as the initial script on the test screen.

- **SET COLS ON/<u>OFF</u>**

  The SET COLS command controls whether XPEDITER/TSO displays a column template when browsing a dataset, ddname, or AA SNAP output. The SET COLS command is maintained across test sessions.

- **SET CONFIRM <u>ON</u>/OFF**

  Whenever the D (Delete) line commands are entered, a confirmation message is displayed that reminds you that breakpoints and keeps are removed. The SET CONFIRM command is maintained across test sessions.

- **SET DATAFIND NEXT/<u>ALL</u>**

  You can indicate whether the next or all data names are found on the FIND *data-name* command. If ALL is specified, a long message indicates the total number found. If NEXT is specified, the cursor moves to the next data name found and highlights it with an informative message in column 74. Note that this SET command is used to override the installed default. The SET DATAFIND command is maintained across test sessions.

- **SET DATETIME <u>ON</u>/OFF**

XPEDITER/TSO checks to see if the date/time stamp of the source listing file and the load module match when loading your program into memory. If the date/time stamp is not in sync, XPEDITER/TSO issues a warning message `DATETIME CONFLICT` and writes the date and time of load module and source listing creation to the log. To resume execution, you can enter SET DATETIME OFF from the primary command area. If source updates were made, however, consequences are unpredictable. Date/time conflicts are detected by XPEDITER/TSO whenever you are setting a qualified breakpoint or whenever the SOURCE command and the INTERCEPT command are issued.

SET DATETIME OFF is overridden with SET DATETIME ON every time you enter the RETEST command or terminate a test session and restart again. You can default to SET DATETIME OFF by editing an INCLUDE script to contain SET DATETIME OFF and specifying the INCLUDE member as the initial script on the test screen.

- **SET DELAY 0/0.*n*/1.*n*/2.*n*/3*n***

  The SET DELAY command, when used with the TRACE command, controls the speed of the execution trace. The default 0 highlights program execution at the real CPU speed. Option 1 slows down the process by one second, 2 by two seconds, and so on. The delay can be set in tenths of a second, where leading and trailing zeros and a trailing decimal point can be omitted. For example, the entries 0.5 and .5 are the same, as are 2.0 and 2. The value can range from 0.0 to 3.9. You must enter the SET DELAY and TRACE commands before you press PF12 or enter GO and execute your program. The SET DELAY command is maintained across test sessions.

- **SET DUMP ON/OFF**

  Controls whether or not a dump is produced. It has no effect on the abend intercept processing within XPEDITER/TSO. If DUMP is set to ON, XPEDITER/TSO then allows a dump to be taken when an abend occurs. A dump will only be produced if one of the standard dump DD files (SYSUDUMP, SYSMDUMP, or SYSABEND) is currently allocated to the test session.

  When requesting a dump, it is best to issue the SET DUMP ON command just prior to execution of the actual code that is leading to the failure so that the size and number of dumps is limited. Correctable or expected conditions may produce unwanted dumps. Once the dump has been obtained, SET DUMP OFF or EXIT the test.

  In the case of an abend that occurs prior to the display of source, it may be necessary to place the SET DUMP ON command in an initial script. Under LE/370, a dump will be taken at the next breakpoint or abend.

  Note: It is also recommended that the ABNLIGNR DD file be allocated to DUMMY so that Abend-AID does not affect the dump contents.

- **SET DYNAMIC** *module-list*

  If the TRACE ALL MODULES command is used, XPEDITER/TSO defaults to tracing statically called modules. The names of dynamically called modules, however, must be specified by the SET DYNAMIC or SET TRANSFER command; otherwise, they are not traced. The SET DYNAMIC command lets you specify up to 100 dynamically called module names, but the command must be in effect before you load the modules in memory. To do this, edit an INCLUDE script to contain SET DYNAMIC *module-name* and specify that INCLUDE member as the initial script on the test screen.

- **SET DYNTRAP ON/OFF**

  If DYNTRAP is set to OFF, XPEDITER/TSO does not intercept dynamically loaded modules, whether they are COBOL or Assembler. If used, the SET DYNTRAP command must be entered within an initial test script that is executed before your program is loaded.

  If a SET DYNAMIC *module-list* command is executed in conjunction with SET DYNTRAP OFF, the specified modules are dynamically loaded. SET DYNTRAP OFF does not override SET DYNAMIC.

- **SET ESPIE ON/OFF**

XPEDITER/TSO activates its own ESPIE (SET ESPIE ON) processing routine to detect S0C1 abend conditions (possible breakpoint) and to handle the condition if it is a breakpoint. A S0C1 abend that is not identified as a breakpoint will then be handled according to the setting of ABNDEXIT (ON or OFF). If needed, the SET DUMP ON command can be used to get a dump of an S0C1 abend, assuming that a dump DD file has been allocated.

Note: If LETRAP is set to ON and the LE environment is active, ESPIE has no effect.

- **SET EXCLUDE** *module-list*

  The XPEDITER/TSO default monitors calls to subroutines. If a subprogram in a subroutine does not use a standard linkage convention to interface with other subprograms, you may experience some difficulty in monitoring the calls. The SET EXCLUDE command lets XPEDITER/TSO treat the system of modules as a black box. In this way, XPEDITER/TSO continues monitoring calls to the specified subroutine, but it no longer keeps track of the calls to the subordinate modules. For example, if a main driver calls a statically linked composite containing an Assembler program that uses a nonstandard interface, the composite load module is excluded.

  The SET EXCLUDE command must be in effect before the module is loaded in memory. To do this, edit an INCLUDE script to contain SET EXCLUDE *module-name* and specify that INCLUDE member as the initial script on the test screen.

- **SET GEN ON/<u>OFF</u>**

  By default, XPEDITER/TSO displays all macros, but not the expanded instructions in the Source screen. You can expand each macro with the GEN command, or expand all macros globally with the SET GEN ON command. The SET GEN ON command must be in effect before you load the program you want expanded. To do this, edit an INCLUDE script to contain SET GEN ON and specify the INCLUDE member as the initial script on the test screen. You can collapse each expanded macro individually with the DELETE command or or globally by entering DELETE GEN or SET GEN OFF. The SET GEN command is maintained across test sessions.

- **SET HEXMODE ON/<u>OFF</u>**

  When HEXMODE is set to ON, any nondisplayable characters resulting from a KEEP or PEEK command entry are represented in hexadecimal format in the log. The hexadecimal characters are presented beneath the character used to depict a nondisplayable character. The SET HEXMODE command affects only the log display. The SET HEXMODE command is maintained across test sessions.

  The SET HEXMODE ON command is equivalent in function to the KEEPH and PEEKH commands. SET HEXMODE ON ensures that sufficient information is provided when an unexpected nonrepresentable character is encountered in a displayed value.

- **SET LANGUAGE <u>ENglish</u>/JApanese**

  The SET LANGUAGE command sets the language that XPEDITER uses for its messages. The initial value is taken from the ISPF language setting.

- **SET LETRAP <u>ON</u>/OFF**

  The SET LETRAP command sets the LE/370 run-time environment TRAP option. SET LETRAP is maintained across test sessions. If LETRAP OFF is specified, TRAP is set to OFF, and XPEDITER’s abend and interrupt usage of ESPIE and ESTAE processing is activated. This prevents the LE/370 handlers and user handlers that rely on ESPIE and ESTAE from getting control. SignaL conditions are unaffected. If the SET LETRAP OFF option is to be used for the current test, it must have been placed in an initial script that was executed at run time or stored in the user profile from a previous test.

- **SET LOG <u>ON</u>/OFF**

  The SET LOG command controls whether the test session audit trail is written to the log file. By default, XPEDITER/TSO commands and their results are written to the log. To prevent the log from becoming full in a test session, you can enter SET LOG OFF. If you edit an INCLUDE script to contain SET LOG OFF and specify that INCLUDE member on the test screen, no entry is made in the log other than the XPEDITER/TSO banner until you enter SET LOG ON.

- **SET LOG** *option* **ON/OFF**

  The SET LOG command controls whether or not the test session audit trail is written to the log file for the following options:

  – AUTOKEEP

  – FIND

  – KEEP

  – MEMORY

  – PEEK

  By default, XPEDITER/TSO commands and their results are written to the log. The LOG FIND option, however, defaults to OFF.

- **SET LOGSIZE 80/132**

  The default maintains an 80-column log file; however, you can override the default and change the log file to a 132-column width by entering SET LOGSIZE 132. Left and right scrolling is available when viewing the log file. If you want to default to SET LOGSIZE 132, edit an INCLUDE script to contain SET LOGSIZE 132 and specify that INCLUDE member as the initial script on the test screen. If you are using a Model 5 terminal, the log file defaults to a width of 132.

- **SET LOWCASE CONVERT/ASIS/NONE/KANA**

  Specifies how lowercase letters found in the values of items are displayed. The options are:

CONVERT	Show lowercase characters from the user source code in uppercase.
ASIS	Display all lowercase characters if they exist. This is the default. Use SET LOWCASE ASIS in conjunction with SET CAPS OFF.
NONE	Show the Source display and all XPEDITER/TSO secondary Source screens in uppercase, allowing no lowercase English characters to be displayed on the screen.
KANA	Does everything option NONE does, as well as specifies all uppercase using the Katakana character translation tables.

  The SET LOWCASE command is maintained across test sessions. This option defaults to KANA if you set your terminal type to 3278KN.

- **SET NOCANCEL** *module-list*

  Excludes listed modules from automatic cancellation. You need to use this setting only if you are using SET AUTOCAN ON to cancel dynamically called modules.

  A module specified with NOCANCEL can contain the wildcard character (*). For example, if you enter SET NOCANCEL ABC*, then modules beginning with the characters ABC are not cancelled.

- **SET NOCOVER** *module-list*

  Allows a list of one to 100 module names which are not to be monitored during a code coverage test. XPEDITER/TSO does not record execution statistics or write any results to a repository for the specified modules.

  Use the SET NOCOVER command when performing a code coverage test with multiple modules to exclude any modules from the code coverage process. This might occur because a module has already been thoroughly tested, statistics are not desired, another user is responsible for testing the module, or the module was coded using non-standard practices or attributes not supported by XPEDITER.

If there are modules within a test which have no corresponding DDIO member allocated to the test, performance can be improved by specifying the SET NOCOVER command for such modules because the DDIO files won't be scanned for the existence of source.

In order to exclude a module from monitoring during a code coverage test, the SET NOCOVER command must be issued before the module is loaded. This can be assured by including the command in an initial script.

The slash ("/") character can be used as a terminating wildcard to cause the match to occur only on the prefix characters preceding the slash. For example:

```
SET NOCOVER PGM12/
```

could be used to exclude from monitoring any module whose name began with the characters PGM12 (such as PGM123, PGM12ABC, PGM12, etc.).

- **SET NOINIT ON/<u>OFF</u>**

  Controls whether a linkage section is initialized to binary zeros or to spaces and zeros when a subroutine is tested stand-alone. This parameter must be used in an initial script.

- **SET NONDISP** <u>.</u>/*char*

  XPEDITER/TSO displays nondisplayable data (non-EBCDIC) as periods for KEEP, KEEPH, PEEK, PEEKH, and MEMORY fields when the data the data is a character. You can change the default character to any character you want by entering the SET NONDISP command. For example, SET NONDISP $ changes nondisplayable characters to dollar signs. Invalid numeric data is displayed with question marks (?), which cannot be changed. The SET NONDISP command is maintained across test sessions.

- **SET PF***nn value*

  There are several ways to alter your XPEDITER/TSO test session PF key settings. You can change your keys by selecting option **3** (XPEDITER/TSO Test Session Keys) from the Defaults Menu (option 0 on the Primary Menu), typing **KEYS** on the primary command line, or using the SET PFnn command. PF keys that are set during a session are maintained across test sessions.

- **SET REFRESH ON/<u>OFF</u>**

  The SET REFRESH command controls whether the entire screen is updated, instead of only the modified line, where output I/O to the terminal is concerned. The default refreshes only the modified portion of the line(s). In dialog mode, the default is ON. The effect of SET REFRESH ON can be observed when XPEDITER/TSO traces your program logic and highlights the source statements as they are executed.

- **SET REVSIZE <u>1M</u>/***nnnn***M/***nnnnnn***K**

  If the MONITOR is set on, then you can use the SET REVSIZE command to set the review log to a size appropriate to your program. The default is 1M. K represents 1024 bytes and M represents 1024K (1048576 bytes). The maximum value allowed is 2047M. It is recommended that you reduce the size only if you have storage constraints above the 16MB line. If REVSIZE is set too low and you use up the available space, the buffer is overwritten and you receive the message `AT OLDEST RETAINED EXECUTION POSITION`.

- **SET RTEREUS <u>ON</u>/OFF**

  At some sites, the COBOL run-time option, RTEREUS, is set to **ON** by default. This may have adverse effects on your XPEDITER/TSO test. For the current test, to set the RTEREUS COBOL run-time option to **OFF**, you must place the SET RTEREUS OFF command in an initial script. The SET RTEREUS ON command allows the default setting of the RTEREUS COBOL run-time option to be used during the test; it **does not** set the RTEREUS COBOL run-time option to **ON**.

- **SET STATIC IGNORE0 <u>ON</u>/OFF**

The SET STATIC IGNORE0 command lets XPEDITER/TSO intercept statically called modules. If STATIC IGNORE0 is set to OFF, XPEDITER/TSO does not automatically intercept static calls.

- **SET TEMPLATE <u>ON</u>/OFF**

  SET TEMPLATE is maintained across test sessions. By default, XPEDITER/TSO displays a column template when displaying data items in both the source and log displays. The template is a dashed line showing column positions, and is helpful for aligning characters during data entry. The length of the column template corresponds to the length of the data area. You can disable this feature with the SET TEMPLATE OFF command. Once issued, any items added to the KEEP window will not have a column template. At any time, you may add a column template to an item in the KEEP window with the **T** line command. You can also remove the column template with the **DT** line command.

  XPEDITER/TSO defaults to displaying a column template on top of the KEEP and PEEK fields for character and zoned data types. This command affects both the log and source displays. The dashed line showing the column position helps you align character and zoned decimal data. The length of the column template corresponds to the length of the data area. SET TEMPLATE OFF removes the column template from the display field. The SET TEMPLATE command is maintained across test sessions.

- **SET TRANSFER** *module-name*

  The XPEDITER/TSO default traces statically called modules when the TRACE command is used; however, it does not trace dynamically called modules unless the module names are specified by the SET DYNAMIC or the SET TRANSFER command. The SET TRANSFER command lets you specify an Assembler interface module that dynamically calls a program, but does not trace the calls to the transfer module itself. The SET TRANSFER command must be in effect before you load the module in memory. You must edit an INCLUDE script that contains SET TRANSFER *module-name* and specify the INCLUDE member as the initial script on the test screen in order to trace the dynamic calls that the transfer module makes before the program begins execution.

- **SET (WINDOW) AUTOKEEP (MAX) <u>0</u>/***nnn*

  This command lets you adjust the size of the Automatic Keep window and determine the placement of the automatically kept items. The variable *n* specifies the number of lines in the window. SET AUTOKEEP 0, the default, displays automatically kept data in the Keep window at the top of the Source display. If you want a separate window for automatically kept items, specify SET AUTOKEEP *n* to set a fixed size Automatic Keep window of *n* lines at the bottom of your source display. If you want a separate window, but want to see as much of your source as possible, you can use the MAX parameter and specify SET AUTOKEEP MAX *n* to set an adjustable size Automatic Keep window at the bottom of the screen that dynamically expands up to *n* lines. This command is retained across test sessions.

- **SET (WINDOW) KEEP (MAX) <u>5</u>/***nnn*

  This command lets you adjust the size of the Keep window. The variable *n* specifies the number of lines in the window. If the MAX parameter is specified, (e.g., SET KEEP MAX *n*), an adjustable size Keep window that dynamically expands up to *n* lines is set. If the MAX parameter is not specified, (e.g., SET KEEP *n*), a fixed size Keep window of *n* lines is set. This command is retained across test sessions.

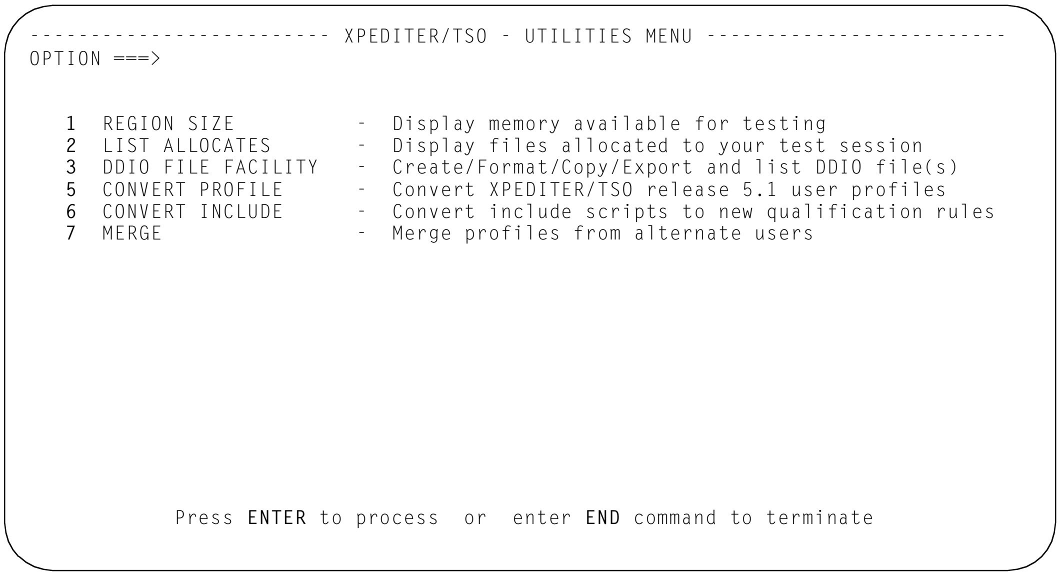
- **SET (WINDOW) SOURCE (MIN)<u>3</u>/***nnn*

  This command lets you set the minimum size of the Source display. The variable *n* specifies the minimum number of lines to display. For example, SET SOURCE 3 sets a minimum of three source lines to be displayed. The Source window will expand as much as possible, depending on the maximum size of the Keep and Automatic Keep windows. This command is retained across test sessions.

Appendix E. XPEDITER/TSO Utilities

The XPEDITER/TSO utility functions are accessed by selecting option **5** on the Primary Menu. The Utilities Menu shown in Figure E-1 is displayed. From this menu, you can select the following functions:

1. Display the memory available for debugging (REGION SIZE).
2. Display the ddnames and dsnames currently allocated to your TSO session (LIST ALLOCATES).
3. Create, browse, delete, lock/unlock, display, and list DDIO files (DDIO FILE FACILITY).
4. Merge profiles from other user IDs (MERGE).

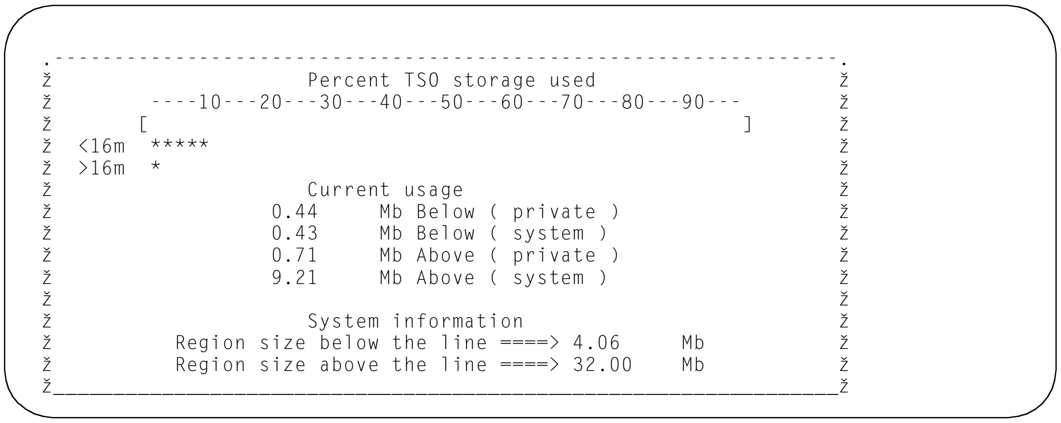


```
------------------------ XPEDITER/TSO - UTILITIES MENU ------------------------
OPTION ===>


   1  REGION SIZE          -  Display memory available for testing
   2  LIST ALLOCATES       -  Display files allocated to your test session
   3  DDIO FILE FACILITY   -  Create/Format/Copy/Export and list DDIO file(s)
   5  CONVERT PROFILE      -  Convert XPEDITER/TSO release 5.1 user profiles
   6  CONVERT INCLUDE      -  Convert include scripts to new qualification rules
   7  MERGE                -  Merge profiles from alternate users












            Press ENTER to process  or  enter END command to terminate
```

Figure E-1. XPEDITER/TSO Utilities Menu

Displaying Available Memory (REGION SIZE)

Selecting option **1** (REGION SIZE) on the Utilities Menu will display an example of a condensed screen of memory and storage specifications available for debugging. This menu screen is similar to the screen displayed when you enter a SHOW REGION command from within an XPEDITER/TSO debugging session. The condensed storage specification screen is displayed in Figure E-2 on page E-2.



```
.-----------------------------------------------------------------.
ž                    Percent TSO storage used                     ž
ž        ----10---20---30---40---50---60---70---80---90---        ž
ž       [                                                  ]      ž
ž  <16m  *****                                                    ž
ž  >16m  *                                                        ž
ž                    Current usage                                ž
ž                 0.44     Mb Below ( private )                   ž
ž                 0.43     Mb Below ( system )                    ž
ž                 0.71     Mb Above ( private )                   ž
ž                 9.21     Mb Above ( system )                    ž
ž                                                                 ž
ž                    System information                           ž
ž          Region size below the line ====> 4.06     Mb           ž
ž          Region size above the line ====> 32.00    Mb           ž
ž_________________________________________________________________ž
```

Figure E-2. XPEDITER/TSO Storage Specifications

The following guidelines give you some idea as to what the memory requirements are for the different XPEDITER/TSO execution environments:

BATCH	1,500,000 bytes
TSO	1,500,000 bytes
IMS	2,000,000 bytes
BTS	2,500,000 bytes
DB2	2,000,000 bytes

If you are using **DATACOM/DB**, **IDMS/DB**, or **System 2000**, the recommended memory is 2,000,000 bytes.

The listed figures are approximate; your actual memory requirements will vary depending on the size of the program being debugged. The amount of memory required by XPEDITER/TSO is related to the number of statements in each program that has a DDIO loaded into memory.

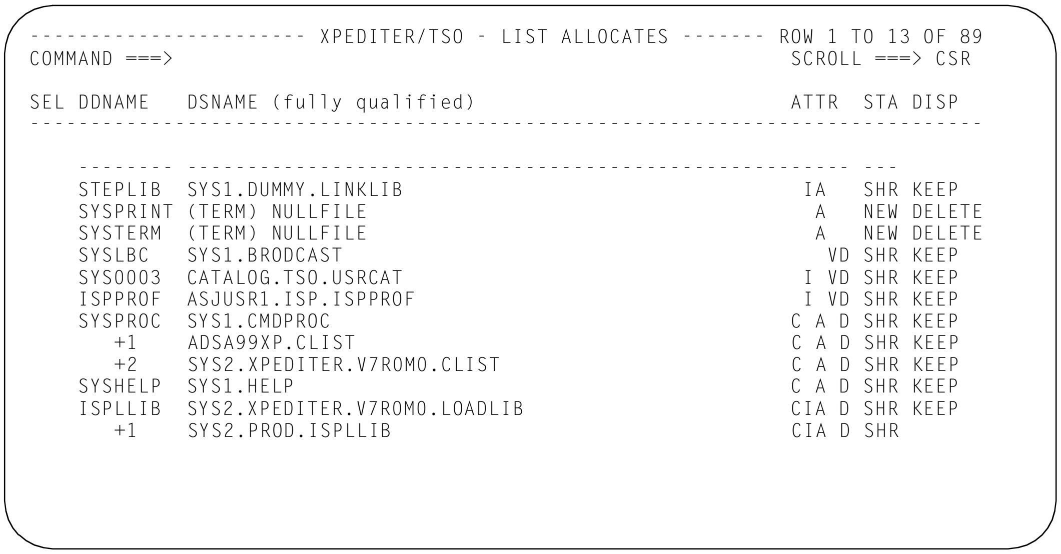
Displaying File Allocations (LIST ALLOCATES)

Option **2** (LIST ALLOCATES) on the Utilities Menu displays the files allocated to your TSO debugging session. The same display can be reached by entering the ALC command from within an XPEDITER/TSO debugging session. The following line commands are available, entered in the SEL column:

A	Concatenation destination (After)
B	Browse individual dataset
C	Concatenation source (Copy)
D	Deconcatenate request
E	Edit individual dataset
S	Dataset status
U	Unallocation request

The SHOW ALLOCATES command displays the same information in essentially the same format, but the line commands are not available.

Figure E-3 on page E-3 is a sample of the output from this function.



```
---------------------- XPEDITER/TSO - LIST ALLOCATES ------- ROW 1 TO 13 OF 89
COMMAND ===>                                                 SCROLL ===> CSR

SEL DDNAME   DSNAME (fully qualified)                        ATTR  STA DISP
-------------------------------------------------------------------------------

    -------- ---------------------------------------------- ---
    STEPLIB  SYS1.DUMMY.LINKLIB                               IA   SHR KEEP
    SYSPRINT (TERM) NULLFILE                                   A   NEW DELETE
    SYSTERM  (TERM) NULLFILE                                   A   NEW DELETE
    SYSLBC   SYS1.BRODCAST                                      VD SHR KEEP
    SYS0003  CATALOG.TSO.USRCAT                               I VD SHR KEEP
    ISPPROF  ASJUSR1.ISP.ISPPROF                              I VD SHR KEEP
    SYSPROC  SYS1.CMDPROC                                    C A D SHR KEEP
       +1    ADSA99XP.CLIST                                  C A D SHR KEEP
       +2    SYS2.XPEDITER.V7R0M0.CLIST                      C A D SHR KEEP
    SYSHELP  SYS1.HELP                                       C A D SHR KEEP
    ISPLLIB  SYS2.XPEDITER.V7R0M0.LOADLIB                    CIA D SHR KEEP
       +1    SYS2.PROD.ISPLLIB                               CIA D SHR
```

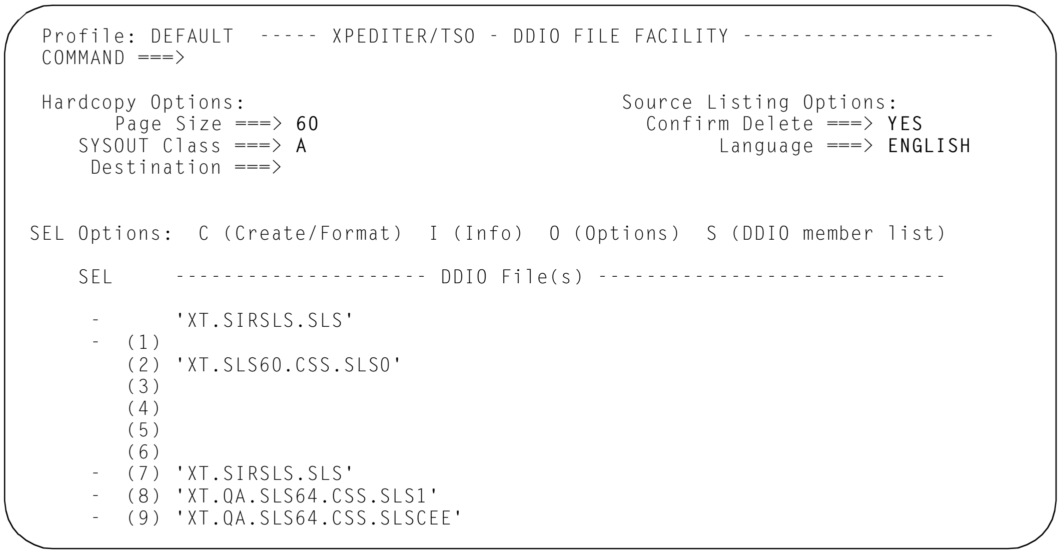
Figure E-3. List Allocates Screen

DDIO File Facility

A DDIO file is required by XPEDITER during compile time to store source listings containing the information necessary for XPEDITER/TSO to reference various elements of your program during testing.

Note: In some cases, a DDIO file was created and formatted during the installation of Compuware's Shared Services. If not, you can use this utility to create a DDIO file.

Option **3** (DDIO FILE FACILITY) on the Utilities Menu enables you to create and format a DDIO file; obtain format information about the file; copy, move, export, or import DDIO files; and obtain a list of DDIO file members and browse, delete, print, lock, or unlock members from that list. When you access the DDIO File Facility, the DDIO File Facility screen shown in Figure E-4 is displayed.



```
 Profile: DEFAULT  ----- XPEDITER/TSO - DDIO FILE FACILITY ---------------------
 COMMAND ===>

 Hardcopy Options:                              Source Listing Options:
       Page Size ===> 60                          Confirm Delete ===> YES
    SYSOUT Class ===> A                                 Language ===> ENGLISH
     Destination ===>


SEL Options:  C (Create/Format)  I (Info)  O (Options)  S (DDIO member list)

    SEL     -------------------- DDIO File(s) ----------------------------

     -       'XT.SIRSLS.SLS'
     -  (1)
        (2) 'XT.SLS60.CSS.SLS0'
        (3)
        (4)
        (5)
        (6)
     -  (7) 'XT.SIRSLS.SLS'
     -  (8) 'XT.QA.SLS64.CSS.SLS1'
     -  (9) 'XT.QA.SLS64.CSS.SLSCEE'
```

Figure E-4. DDIO File Facility Screen

The options on this screen are defined as follows:

Page Size
Number of lines per page for the printed hardcopy. Valid entry is any number between 20 and 99.

SYSOUT Class
JES spool class that prints the output.

Destination
JES destination code for the output.

Confirm Delete
Specify **YES** to display a confirmation panel before deleting the report.

Language
Specify the language for online and hardcopy translation:

ENGLISH	Provides mixed-case English support.
JAPANESE	Provides Japanese Kanji support on DBCS terminals.

SEL Options
Type one of the following SEL options next to the DDIO file:

C To create and format a DDIO file. The DDIO Create/Format Facility screen shown in Figure E-5 is displayed.

I To display format information for the specified file. The DDIO File Information screen shown in Figure E-9 on page E-9 is displayed.

O To copy, move, import, or export the specified DDIO file. The DDIO Options Facility screen shown in Figure E-7 on page E-6 is displayed.

S To browse, delete, print, and lock/unlock a specified DDIO file. The Source Listing Dataset Directory screen shown in Figure E-8 on page E-8 is displayed.

DDIO File(s)
Enter a dataset name on the dashed lined below this field to access a DDIO file not in the list or to create a new DDIO file.

Creating and Formatting a DDIO File

To create and format a DDIO file, enter an uncataloged dataset name on the dashed line beneath the DDIO File(s) field on the DDIO File Facility screen. Then enter a **C** in the SEL area to display the DDIO Create/Format Facility screen shown in Figure E-5. This screen is used to enter the information necessary to create and format the DDIO file.

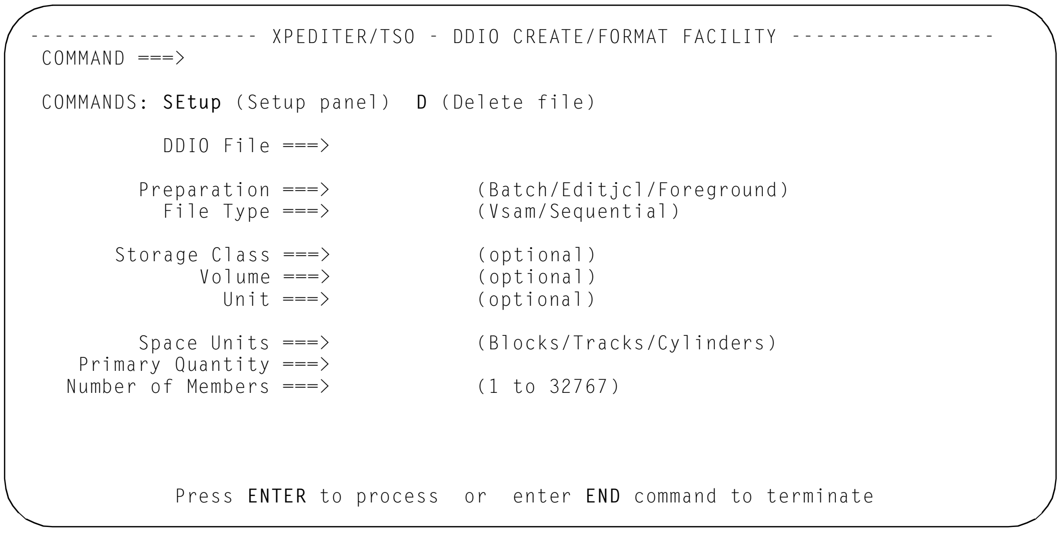

```
------------------- XPEDITER/TSO - DDIO CREATE/FORMAT FACILITY -----------------
 COMMAND ===>

 COMMANDS: SEtup (Setup panel)  D (Delete file)

           DDIO File ===>

         Preparation ===>            (Batch/Editjcl/Foreground)
           File Type ===>            (Vsam/Sequential)

       Storage Class ===>            (optional)
              Volume ===>            (optional)
                Unit ===>            (optional)

         Space Units ===>            (Blocks/Tracks/Cylinders)
    Primary Quantity ===>
   Number of Members ===>            (1 to 32767)



            Press ENTER to process  or  enter END command to terminate
```

Figure E-5. DDIO Create/Format Facility Screen

The options on this screen are defined as follows:

DDIO File
The new uncataloged dataset name that was entered on the DDIO File Facility screen is prefilled in this field.

Preparation
Enter the processing environment:

Batch	The default. Builds and submits the JCL.
Editjcl	Builds the JCL and lets you view it before submitting it. You can submit the JCL from the screen.
Foreground	Submits the job and runs it in foreground.

File Type
Enter the file type: VSAM or Sequential. VSAM is the default. There are no external functional differences in the choice specified.

Storage Class
Enter a valid storage allocation attribute value for the Storage Management Subsystem (SMS).

Volume
Optionally, enter the volume serial number.

Unit
Optionally, enter the unit name for the new dataset.

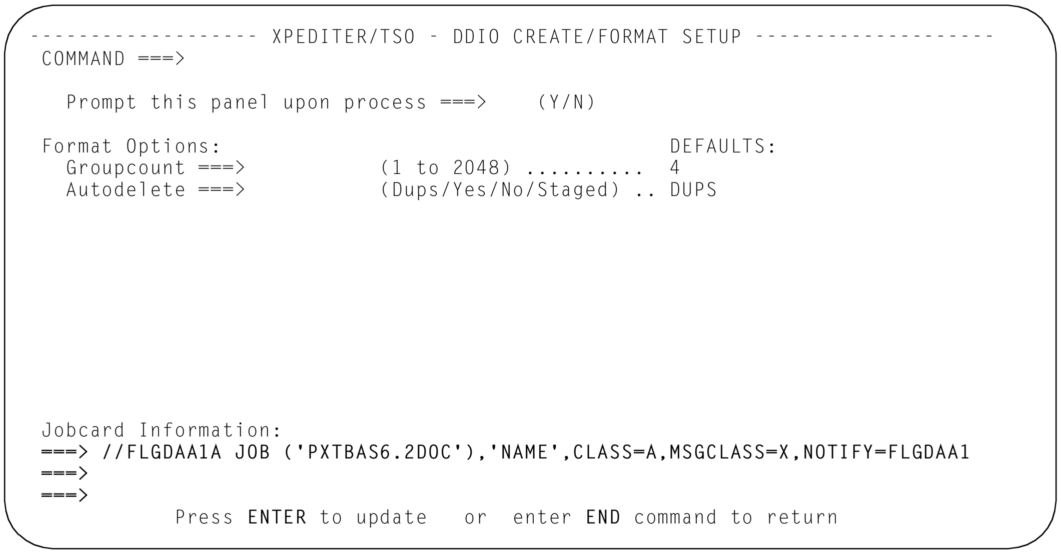
Space Units
Enter space allocation units (blocks, tracks, or cylinders) associated with the primary quantity.

Primary Quantity
Enter the amount of space to be allocated for the entire file.

Number of Members
Enter the total number of members the file can hold.

Specifying Setup Options to Create a DDIO File

To specify setup options for creating a DDIO file, enter the **SETUP** command on the DDIO Create/Format Facility screen. The DDIO Create/Format Setup screen shown in Figure E-6 is displayed.



```
------------------ XPEDITER/TSO - DDIO CREATE/FORMAT SETUP --------------------
 COMMAND ===>

   Prompt this panel upon process ===>    (Y/N)

 Format Options:                                     DEFAULTS:
   Groupcount ===>           (1 to 2048) ..........  4
   Autodelete ===>           (Dups/Yes/No/Staged) .. DUPS









 Jobcard Information:
 ===> //FLGDAA1A JOB ('PXTBAS6.2DOC'),'NAME',CLASS=A,MSGCLASS=X,NOTIFY=FLGDAA1
 ===>
 ===>
            Press ENTER to update   or  enter END command to return
```

Figure E-6. DDIO Create/Format Setup Screen

The options on this screen are defined as follows:

Prompt this panel upon process
If Y (YES) is entered, this screen is automatically displayed before the job is submitted. Enter N (NO) if you do not want to automatically display this screen.

Groupcount
The number of blocks allocated to a DDIO member each time additional space is needed. The default is 4.

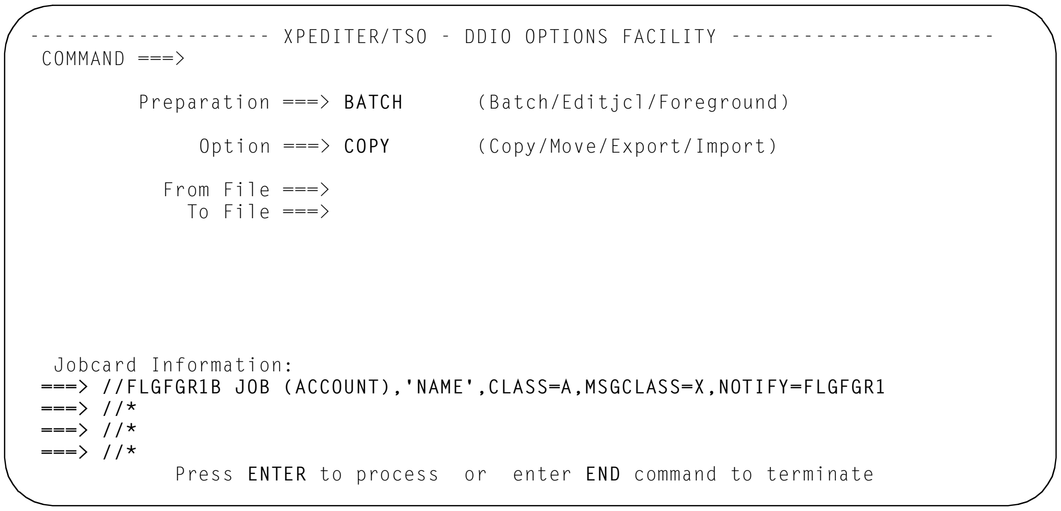
Autodelete
Enter one of the following:

Dups	Ensures that at least one version of every member remains in the file. Dups is recommended for XPEDITER.
Yes	Deletes the oldest unlocked member in the file if more space is required to add new members.
No	If the DDIO file becomes full, you must manually delete members from the file.
Staged	Ensures that at least one version of every member remains in the file, except when all members are locked. The oldest auto-locked member not in use is deleted.

Jobcard Information
Enter job statements for batch and editjcl preparations.

Copying, Moving, Exporting, and Importing DDIO Members

To copy, move, export, and import DDIO members, enter the **O** (Options) line command next to the DDIO file on the DDIO File Facility screen. When you press **Enter**, the DDIO Options Facility screen shown in Figure E-7 is displayed.



```
------------------- XPEDITER/TSO - DDIO OPTIONS FACILITY ---------------------
COMMAND ===>

        Preparation ===> BATCH      (Batch/Editjcl/Foreground)

             Option ===> COPY       (Copy/Move/Export/Import)

          From File ===>
            To File ===>





 Jobcard Information:
===> //FLGFGR1B JOB (ACCOUNT),'NAME',CLASS=A,MSGCLASS=X,NOTIFY=FLGFGR1
===> //*
===> //*
===> //*
           Press ENTER to process  or  enter END command to terminate
```

Figure E-7. DDIO Options Facility Screen

The options on this screen are:

Preparation
Enter the processing environment:

Batch	The default. Builds and submits the JCL.
Editjcl	Builds the JCL and lets you view it before submitting it. You can submit the JCL from the screen.
Foreground	Submits the job and runs it in foreground.

Options
Enter one of the following options:

Copy	To copy members from one DDIO file to another DDIO file.
Move	To move members from one DDIO file to another DDIO file.
Export	To copy members from a DDIO file to a sequential dataset, which can later be restored into the same or another DDIO file using the Import option.
Import	To restore previously exported DDIO file members from a sequential dataset.

From File
Specify either the input DDIO file for the Copy, Move, and Export options, or the input sequential dataset for the Import option.

Note: The member name or pattern can also be specified with the input dataset.

To File
Specify either the output DDIO file for the Copy, Move, and Import options, or the output sequential dataset for the Export option.

Jobcard Information
Enter job statements for batch and editjcl preparations.

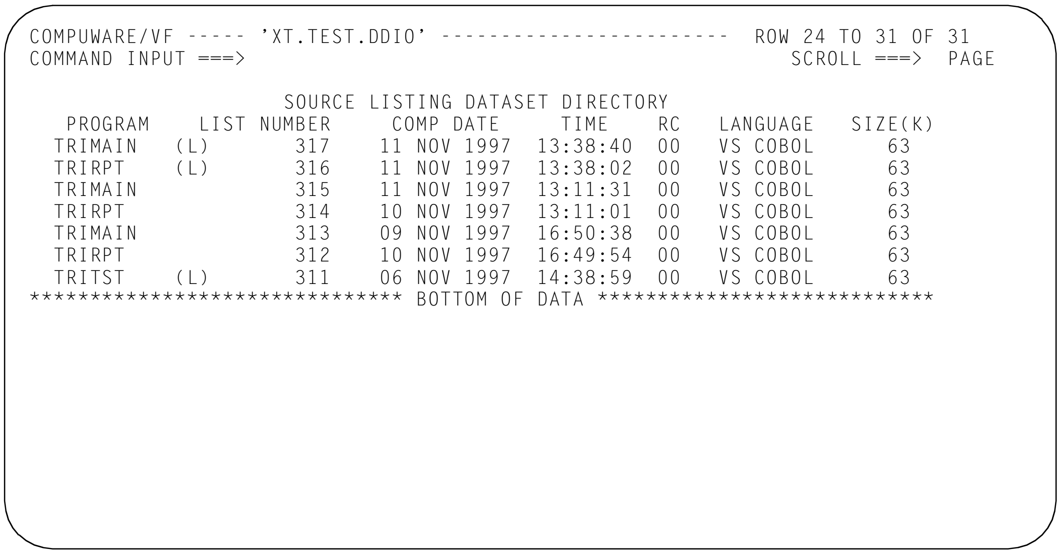
Browsing, Deleting, Printing, Locking, and Unlocking DDIO Members

To browse, print, delete, lock, and unlock members, enter the **S** (Select) line command next to the DDIO file on the DDIO File Facility screen. When printing a member, also complete the Hardcopy Options shown at the top of the DDIO File Facility screen. When you press **Enter**, a Dataset Directory screen similar to the one shown in Figure E-8 on page E-8 is displayed. This is a Compuware Viewing Facility screen and is referred to as a Source Listing Dataset Directory screen. Additional information on Dataset Directory screens is contained in the Compuware Viewing Facility chapter of the *Compuware Shared Services Installation and Customization Guide*.

The Source Listing Dataset Directory screen displays the names of the programs that are compiled and have had the source listing written to the DDIO. The listings are shown in order from the most to least current listing. Use the PF7 and PF8 scrolling keys, and normal user commands to look at additional entries if more than one screen of data exists.

To select a source listing to browse, print, delete, lock, or unlock, position the cursor on the line where the listing occurs and enter a valid line command listed below. Multiple line commands can be entered.

S (Select)	Selects a listing for display
P (Print)	Prints a listing using the hardcopy facility
D (Delete)	Deletes a listing from the DDIO dataset
L (Lock)	Locks a listing so it cannot be automatically deleted
U (Unlock)	Unlocks a listing so it can be automatically deleted



```
COMPUWARE/VF ----- 'XT.TEST.DDIO' -----------------------  ROW 24 TO 31 OF 31
COMMAND INPUT ===>                                            SCROLL ===>  PAGE

                     SOURCE LISTING DATASET DIRECTORY
   PROGRAM    LIST NUMBER     COMP DATE     TIME    RC   LANGUAGE   SIZE(K)
  TRIMAIN   (L)       317    11 NOV 1997  13:38:40  00   VS COBOL      63
  TRIRPT    (L)       316    11 NOV 1997  13:38:02  00   VS COBOL      63
  TRIMAIN             315    11 NOV 1997  13:11:31  00   VS COBOL      63
  TRIRPT              314    10 NOV 1997  13:11:01  00   VS COBOL      63
  TRIMAIN             313    09 NOV 1997  16:50:38  00   VS COBOL      63
  TRIRPT              312    10 NOV 1997  16:49:54  00   VS COBOL      63
  TRITST    (L)       311    06 NOV 1997  14:38:59  00   VS COBOL      63
****************************** BOTTOM OF DATA ***************************
```

Figure E-8. Source Listing Dataset Directory Screen

The Source Listing Dataset Directory screen displays the following information:

PROGRAM
Name of the program. A 3-character field appears after the program name. The system may display one of the following in this field:

(L) Report is locked

***I** Report is incomplete

***IL** Report is incomplete and locked

LIST NUMBER
Number assigned by Compuware postprocessor to identify the listing. Numbering occurs sequentially from 1 until the DDIO file is reformatted.

COMP DATE
Specifies the date the listing was created.

TIME
Specifies the time the listing was created.

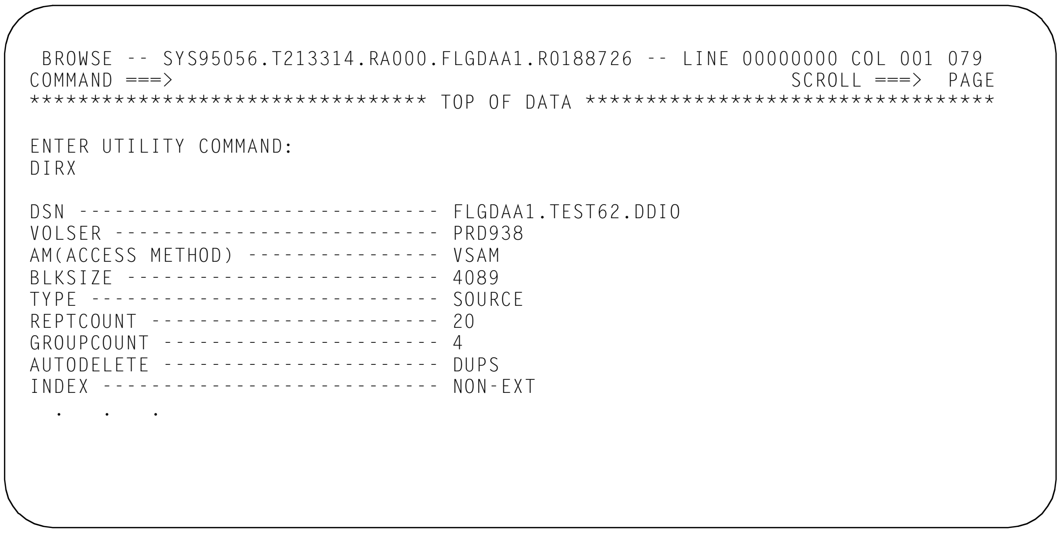
RC
Return code provided from COBOL postprocessor steps when writing to the file. A value of **NA** in this field indicates that a return code is not available because the step ended prematurely due to a time-out or job cancellation.

LANGUAGE
Specifies the program language of the listing.

SIZE
Number of bytes the program is using on the listing file. The listing is compressed as it is written to the file. Do not interpret this byte size as the size of the load module or the size of the source program.

Displaying Format Information for a DDIO File

To display format information about a DDIO file, enter the **I** (Information) line command next to the DDIO file on the DDIO File Facility screen. The screen shown in Figure E-9 on page E-9 is displayed.



```
 BROWSE -- SYS95056.T213314.RA000.FLGDAA1.R0188726 -- LINE 00000000 COL 001 079
COMMAND ===>                                                   SCROLL ===>  PAGE
********************************* TOP OF DATA **********************************

ENTER UTILITY COMMAND:
DIRX

DSN ----------------------------- FLGDAA1.TEST62.DDIO
VOLSER -------------------------- PRD938
AM(ACCESS METHOD) ---------------- VSAM
BLKSIZE ------------------------- 4089
TYPE ---------------------------- SOURCE
REPTCOUNT ----------------------- 20
GROUPCOUNT ---------------------- 4
AUTODELETE ---------------------- DUPS
INDEX --------------------------- NON-EXT
  .   .   .
```

Figure E-9. DDIO File Information Screen

CONVERT PROFILE

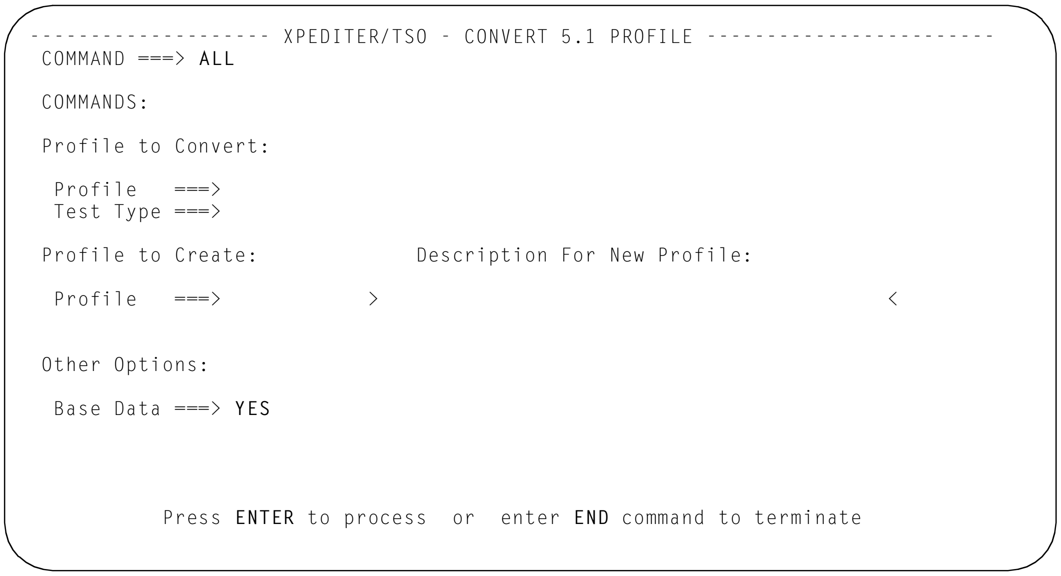
When you select option **5** on the Utilities Menu, the CONVERT PROFILE Release 5.1 screen is displayed.

This utility converts XPEDITER/TSO Release 5.1 user profiles to current profile handling methodology. This will allow compatibility for future releases.

Enter **ALL** on the command line to convert all Release 5.1 profiles or enter the following for selected profiles:

```
Profile to Convert:

  Profile   ===>  Enter the number (0 - 9) of the prefix to be copied.
  Test Type ===>  Enter the number (1 - 11) of the testing environment.

Profile to Create:

  Profile   ===>  Enter the name of the new profile being created.

Other Options:

  Base Data ===>  YES or NO to copy basic user data (color, JOB cards, etc.).
```

The CONVERT PROFILE Release 5.1 screen (Figure E-10 on page E-10) is displayed on the following page:



```
------------------- XPEDITER/TSO - CONVERT 5.1 PROFILE -----------------------
COMMAND ===> ALL

COMMANDS:

Profile to Convert:

 Profile   ===>
 Test Type ===>

Profile to Create:            Description For New Profile:

 Profile   ===>           >                                          <


Other Options:

 Base Data ===> YES



         Press ENTER to process  or  enter END command to terminate
```

Figure E-10. CONVERT PROFILE Release 5.1 Screen

CONVERT INCLUDE

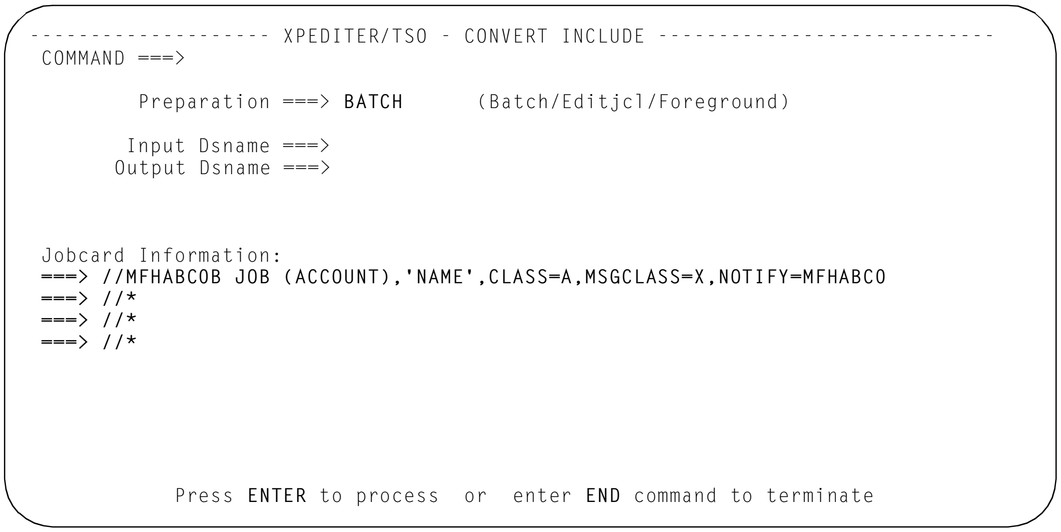
When you select option **6** on the Utilities Menu, the CONVERT INCLUDE screen is displayed.

Use this utility to convert INCLUDE SCRIPT dataset members (created prior to XPEDITER/TSO Release 6.3) to the new qualification rule. All references of delimiter '.' used for module qualification will be changed to ':'.

The following examples use **TRITST** for module qualification:

```
         Before Conversion              After Conversion
         -----------------------        ----------------------
         KEEP  TRITST.TST-REC           KEEP  TRITST:TST-REC
         SKIP  TRITST.18                SKIP  TRITST:18
         AFTER TRITST.                  AFTER TRITST:

       Preparation ===> Specify environment for which processing is to take place.

      Input Dsname ===> Specify an INCLUDE SCRIPT dataset to convert.
     Output Dsname ===> Specify a cataloged partitioned dataset to copy
                        converted INCLUDE SCRIPT members to.  The Dsname
                        may also be the same as the Input Dsname, thus
                        updating original members if so desired.

     Jobcard Information: Specify Job statements for BATCH and EDITJCL preparations.
```

The CONVERT INCLUDE screen (Figure E-11) is displayed below:

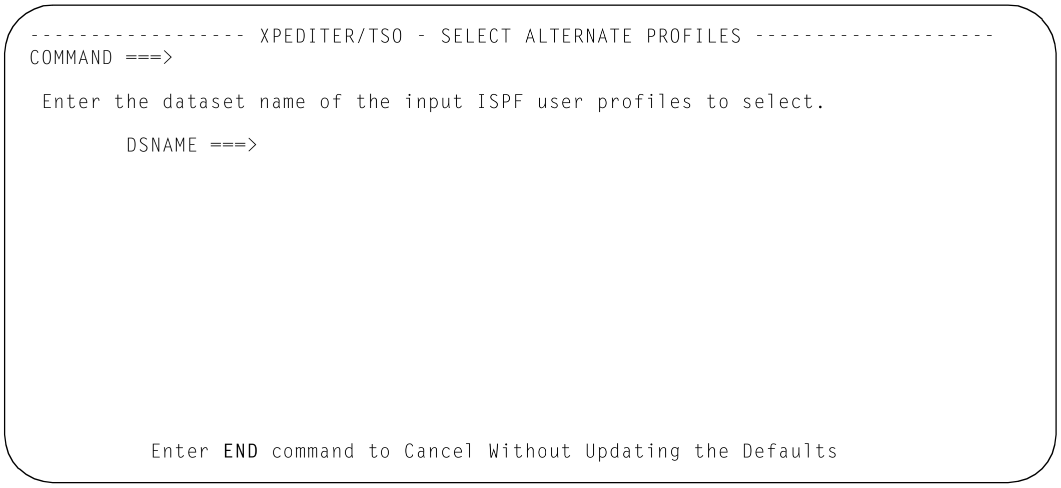


```
-------------------- XPEDITER/TSO - CONVERT INCLUDE ----------------------------
 COMMAND ===>

         Preparation ===> BATCH      (Batch/Editjcl/Foreground)

        Input Dsname ===>
       Output Dsname ===>



 Jobcard Information:
 ===> //MFHABCOB JOB (ACCOUNT),'NAME',CLASS=A,MSGCLASS=X,NOTIFY=MFHABCO
 ===> //*
 ===> //*
 ===> //*






            Press ENTER to process  or  enter END command to terminate
```

Figure E-11. CONVERT INCLUDE Screen

Selecting Alternate Profiles (MERGE PROFILE)

Option **7** on the Utilities Menu merges profiles from other user IDs into your personal profile. The Select Alternate Profiles screen shown in Figure E-12 is displayed below.



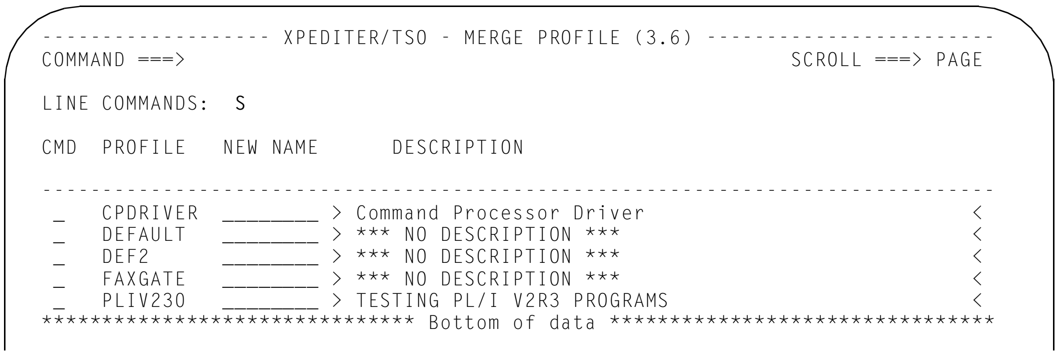
```
----------------- XPEDITER/TSO - SELECT ALTERNATE PROFILES --------------------
COMMAND ===>

 Enter the dataset name of the input ISPF user profiles to select.

        DSNAME ===>














          Enter END command to Cancel Without Updating the Defaults
```

Figure E-12. Select Alternate Profiles Screen

The option on this screen is:

DSNAME
: Enter the name of the ISPF profile dataset in the DSNAME field to identify the source of the XPEDITER/TSO profiles. Entering a dataset name results in the display of the Merge Profile screen—a scrollable list from which you can make your selection.

After you have typed the name of the ISPF profile dataset in the DSNAME field, press **ENTER** to display the Merge Profile screen.



```
------------------- XPEDITER/TSO - MERGE PROFILE (3.6) ------------------------
COMMAND ===>                                                   SCROLL ===> PAGE

LINE COMMANDS:  S

CMD  PROFILE   NEW NAME      DESCRIPTION

-------------------------------------------------------------------------------
 _   CPDRIVER  ________ > Command Processor Driver                            <
 _   DEFAULT   ________ > *** NO DESCRIPTION ***                              <
 _   DEF2      ________ > *** NO DESCRIPTION ***                              <
 _   FAXGATE   ________ > *** NO DESCRIPTION ***                              <
 _   PLIV230   ________ > TESTING PL/I V2R3 PROGRAMS                          <
****************************** Bottom of data *********************************
```

Figure E-13. MERGE PROFILE Screen

The list provided on the Merge Profile screen (Figure E-13) displayed above shows the alternate profiles that exist in another user's XPEDITER profile table. To merge one or more of these profiles into your profile, use the **S**elect command. By using SELECT, you can change the profile name and/or description by simply typing new values.

Appendix F. Binding the Application Plan

If the XPEDITER for DB2 Extension and File-AID for DB2 are installed at your site, and you want to use either product during the debugging session, you must bind your program application plan with the File-AID for DB2 DBRM if any of the following apply:

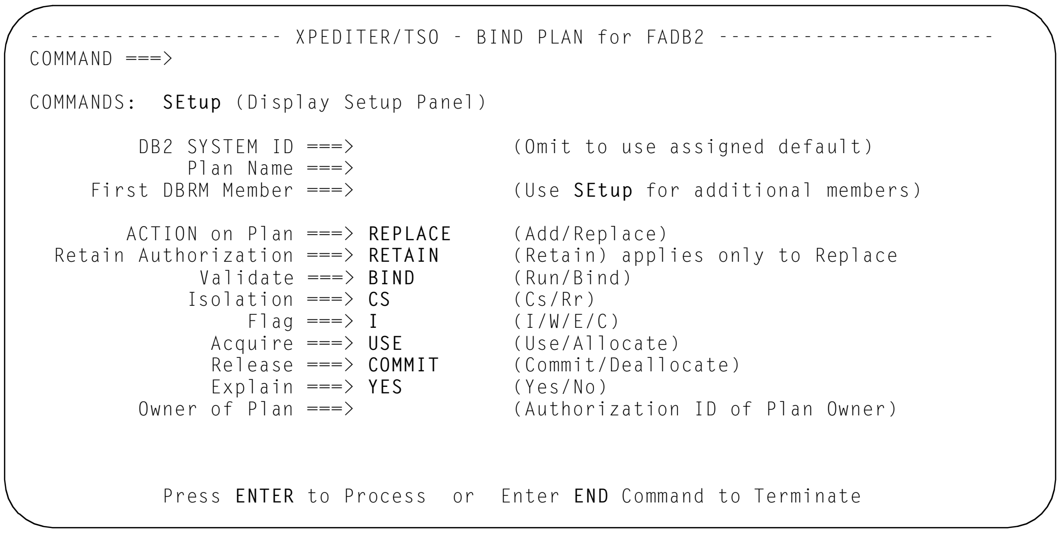
1. The program executes statically compiled SQL statements. If the program executes only dynamically inserted SQL statements, you can either bind your application plan with File-AID for DB2 or use the File-AID for DB2 default plan in place of your application plan.
2. The program is executed in the IMS environment.

Bind File-AID DB2 Program Plan Screen

Select option **1** (PREPARE) from the Primary Menu to display the Program Preparation Menu. Then, select option **3** (BIND) from the Program Preparation Menu to display the screen shown in Figure F-1. To perform the bind function, you must have DB2 authority.

The valid commands on the screen are:

SETUP	Displays the DBRM Members and DBRM Libraries screen, which is used to specify the DBRM members and datasets you want to use as input to the bind function. Refer to "DBRM Libraries Screen" on page F-3 for a description of this screen.
RESTORE	Reinstates all installed default values (on all setup and installation screens) to the values assigned during installation.



```
-------------------- XPEDITER/TSO - BIND PLAN for FADB2 ----------------------
COMMAND ===>

COMMANDS:  SEtup (Display Setup Panel)

         DB2 SYSTEM ID ===>             (Omit to use assigned default)
             Plan Name ===>
     First DBRM Member ===>             (Use SEtup for additional members)

        ACTION on Plan ===> REPLACE     (Add/Replace)
  Retain Authorization ===> RETAIN      (Retain) applies only to Replace
              Validate ===> BIND        (Run/Bind)
             Isolation ===> CS          (Cs/Rr)
                  Flag ===> I           (I/W/E/C)
               Acquire ===> USE         (Use/Allocate)
               Release ===> COMMIT      (Commit/Deallocate)
               Explain ===> YES         (Yes/No)
         Owner of Plan ===>             (Authorization ID of Plan Owner)



            Press ENTER to Process  or  Enter END Command to Terminate
```

Figure F-1. Bind Plan for FADB2 Screen

The values shown in Figure F-1 are the recommended values, which will provide better performance over the other listed options.

When you press **Enter** from this screen, the bind operation begins. You are binding the File-AID for DB2 DBRM members with the DBRM member(s) you list in the First DBRM

Member field and on the DBRM Members and DBRM Libraries screen (described in "DBRM Libraries Screen" on page F-3). You do not have to specify the File-AID for DB2 DBRM members; they are automatically specified by the bind function.

The options on the Bind Plan for FADB2 screen are:

DB2 SYSTEM ID

The DB2 subsystem name (SSN). Specify the proper SSN to use for testing.

If the DB2 system ID is omitted, the system name specified during the IBM/DB2 installation is used. If errors occur, refer to your systems programmer for the appropriate SSN.

Your plan is saved in the catalog of the specified subsystem.

Plan Name

Required. Specify the DB2 application plan to be used by DB2 to process SQL statements encountered during program execution. The plan name must follow the DB2 defined syntax for plan names. Refer to the *IBM DB2 Reference Manual* for the correct syntax.

A DB2 application plan is the output from the bind function, which converts the output from the DB2 precompiler to a usable control structure called an application plan. During this process, access paths to the data are selected and some authorization checking is performed. The plan is used by DB2 to process SQL statements encountered during program execution.

First DBRM Member

Required. Specify the first DBRM to be included in the bind process. If you need to specify additional DBRM members, enter the **SETUP** command to access the DBRM Members and DBRM Libraries screen.

ACTION on Plan

Specify whether the plan is a new plan (Add) or if it is to Replace another plan by the same name.

Retain Authorization

Use **Retain** if you specified Replace in the ACTION on Plan field. Otherwise, leave this field blank.

Validate

Use **Bind**. For performance reasons, the validation of the plan should be done during the bind process.

Isolation

Use **CS** for better performance and reduced lockout situations. Use **RR** only if you want to retain locks on read pages until you reach a commit (synchronization) point.

Flag

Indicates the level of messages you want to see resulting from the bind. Specify one of the following options:

I	All messages
W	Warning, error, and completion messages
E	Only error and completion messages
C	Only completion messages

Acquire

Specify one of the following options:

Use	To allocate DB2 resources when and as needed.
Allocate	To allocate the resources at the time the plan is allocated.

Release

Specify one of the following options:

Commit	To free up the resources after a commit is successfully executed.
Deallocate	To deallocate any resources when the plan is deallocated.

Explain

Specify **Yes** to obtain explain information at bind time. If you specify Yes, you must have access to a Plan_Table in which DB2 can store the information relating to the SQL calls in your program.

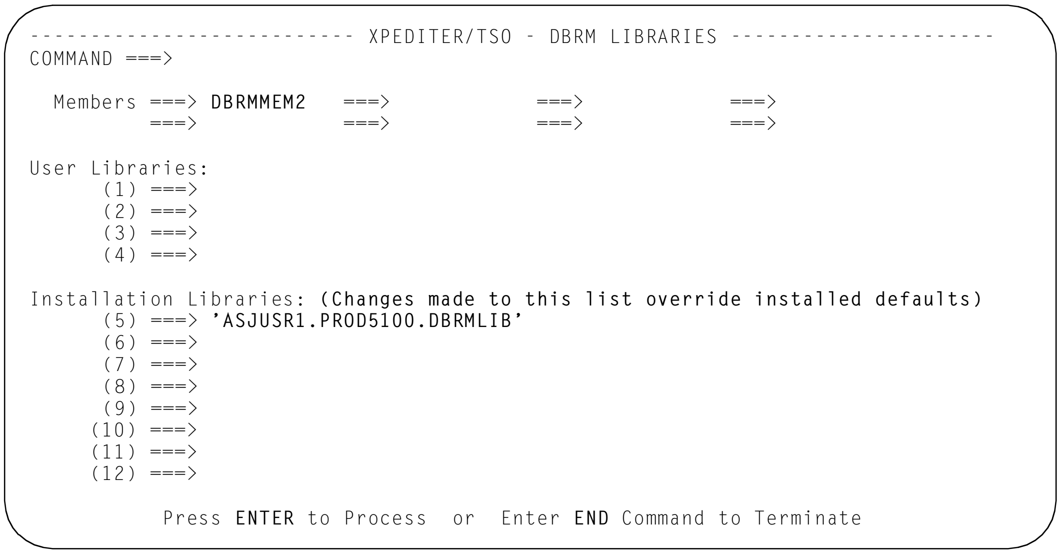
You can allocate a Plan_Table for yourself or use a common Plan_Table.

Owner of Plan

If you want to indicate an owner other than yourself, enter the authorization ID of that person. If omitted, you are, by default, the owner of the plan.

DBRM Libraries Screen

When you enter **SETUP** on the Bind Plan for FADB2 screen, the screen shown in Figure F-2 appears. The DBRM Libraries screen is used to enter the names of the DBRM members and DBRM libraries to be used to bind a DB2 plan. If no DBRM members are entered on this screen, a plan is created that contains the File-AID for DB2 DBRM members and the DBRM entered in the First DBRM Member field on the Bind Plan for FADB2 screen.



```
-------------------------- XPEDITER/TSO - DBRM LIBRARIES ----------------------
COMMAND ===>

  Members ===> DBRMMEM2   ===>            ===>            ===>
          ===>            ===>            ===>            ===>

User Libraries:
      (1) ===>
      (2) ===>
      (3) ===>
      (4) ===>

Installation Libraries: (Changes made to this list override installed defaults)
      (5) ===> 'ASJUSR1.PROD5100.DBRMLIB'
      (6) ===>
      (7) ===>
      (8) ===>
      (9) ===>
     (10) ===>
     (11) ===>
     (12) ===>

         Press ENTER to Process  or  Enter END Command to Terminate
```

Figure F-2. DBRM Members and DBRM Libraries Screen

The options on this screen are:

Members

You can specify one to eight additional DBRM modules to be combined with the File-AID for DB2 DBRM members when creating DB2 plans. The DBRM modules named in these fields must be members of the DBRM libraries defined below.

A DBRM module member is one output of a DB2 compile preprocessor. DBRM modules are combined by the bind process to create plans that are used by DB2 for processing of program SQL statements. The File-AID for DB2 DBRMs are automatically included in the bind process, but do not appear in the member list.

User Libraries

Required only if the DBRM members specified are not contained in the list of installation libraries. You can specify up to four libraries. Normal concatenation rules are in effect—libraries specified first are searched first, and buffer space for the library list is determined by the first library specified.

Installation Libraries

The installer entered the dsnames of common DBRM libraries that should be allocated to any sessions. These libraries are included in the DBRM library list for a plan that is created using the bind process. You can override the installed default libraries by specifying new libraries in these fields.

Appendix G. DBCS Support

XPEDITER/TSO and XPEDITER/IMS lets you use the single-byte character set (SBCS), the double-byte character set (DBCS), and mixed (a combination of DBCS and SBCS) data streams in your application programs and XPEDITER/TSO and XPEDITER/IMS screen functions, testing functions, and data manipulation during a test session.

The attributes of the SBCS, DBCS, and mixed data streams are as follows:

SBCS An alphanumeric character set that lets one EBCDIC character occupy one byte. It is used for languages that use 1-byte characters.

DBCS A graphic character set that lets 1 character occupy two bytes. It is used for languages, such as Japanese, that use ideographic characters that cannot be represented in one byte.

Mixed Data streams consisting of SBCS and DBCS characters. Special delimiters are used to identify DBCS data in a mixed field. They are:

- Shift-out (X'0E', called SO) to show the beginning of DBCS data
- Shift-in (X'0F', called SI) to show the end of DBCS data

SO and SI are automatically inserted when you change shift status on the terminal to begin and end DBCS data. SO and SI must always be paired in a mixed data stream and are only displayed by using the appropriate control keys on your terminal.

While most XPEDITER functions operate normally with DBCS data streams, there are some differences in how these fields are scrolled and how data is manipulated. This section describes those differences.

Terminal Support

DBCS is supported only on terminal types where the ISPF variable ZDBCS is set to Yes.

When a test session is invoked, XPEDITER checks the terminal type and switches the character set translation appropriately to one of the following:

- EBCDIC English (no Katakana)
- EBCDIC Katakana (no lowercase English)
- EBCDIC English with DBCS (lowercase English and SBCS Katakana coexist)
- EBCDIC Katakana with DBCS (no lowercase SBCS English)

EBCDIC English and EBCDIC Katakana character sets are determined by an inquiry to the ISPF terminal type default. DBCS capability is determined by an inquiry to the ISPF ZDBCS variable.

When a Katakana (Japanese) terminal is detected, all lowercase English output strings (messages, title, and text) that appear on XPEDITER screens are converted to uppercase English. DBCS lowercase characters are not affected.

CAUTION:

Do not change the SET LOWCASE value if the terminal type is 32xxKN. The results are unpredictable.

Program Support

DBCS is supported by XPEDITER in interactive mode only. DBCS can be used in standard TSO, including IMS/DB and BTS, IMS/DC (MPP, BMP, IFP), and ISPF dialog programs.

Note: DBCS is not supported in batch connect and native TSO.

DBCS Fields on the Source Display

The following table describes the areas on the XPEDITER Source display screen where DBCS data is valid. The table indicates whether the area is an input or output area, the field attributes, and a description of what the data consists of.

DBCS user-defined words may contain both double-byte EBCDIC and double-byte non-EBCDIC characters.

Screen Area	I/O	Attribute	Description
Header line	O	Mixed	Displays SBCS and DBCS in the screen title.
Primary command line	I/O	Mixed	Enter SBCS English XPEDITER commands. Enter SBCS or DBCS data names, paragraph names, and literals.
Message line	O	Mixed	Displays SBCS and DBCS in messages.
Source area	O	Mixed	Displays SBCS program words, SBCS or DBCS user-defined words (data names, paragraph names, literals, and comments).
Inserted lines	I/O	Mixed	Enter SBCS or DBCS user-defined words (data names, paragraph names, and literals).
Keep/Peek windows	I/O	Mixed	• Alphanumeric fields: Displays and allows entry of DBCS or SBCS data. • Graphic fields: Displays and allows entry of DBCS data only.
Log and Show screens	O	Mixed	Displays DBCS and SBCS data names, paragraph names, literals, and source text.

Table G-1. DBCS Fields on the XPEDITER Source Display Screen.

Character set validation is based on field type (attribute) and terminal type. DBCS data in a mixed field is validated using the following criteria:

1. All occurrences of SO are paired with SI (field level check).
2. All DBCS character lengths are even (field level check)
3. Each byte of a DBCS character is within a valid range X'41' to X'FE' and the DBCS blank (X'4040').

If 1 and 2 are not met, the DBCS field is converted to SBCS representation. If 3 is not met, the DBCS character is shown as a nondisplayable character (X'4195').

CAUTION:

Opening a Keep or Peek window on a group-level variable in which any of the subordinate items have DBCS data will display indistinguishable characters in the window. XPEDITER treats the kept group-level variable as an alphanumeric field and converts the entire window to SBCS representation. If all of the subordinate items have SBCS data only, it is acceptable to open a Keep or Peek window on the group-level variable.

Scrolling DBCS Data

When the PEEK or KEEP command is used, the data is displayed in fully scrollable windows. 30 bytes are displayed at a time. Data exceeding the 30 bytes limit can be displayed by scrolling left or right using the DLEFT or DRIGHT commands or by using the (,(n,),)n, and :n line commands.

Scrolling is always by byte, not by character. When scrolling mixed or DBCS fields, the display window will be adjusted when the result will split a DBCS character. The basic rule is that if the first byte of a DBCS character is in the window at the left or right boundaries, the complete character is displayed. The right boundary is extended by one byte to display the character. If the second byte of a DBCS character is in the left boundary of the window, the field is truncated by one byte on the left and the character is not displayed.

When a mixed field is truncated, XPEDITER inserts and deletes the SO and SI delimiters as needed to maintain the integrity of the SO/SI pairing. These inserted SO/SI delimiters are for display purposes only and are not saved in the underlying data. When a DBCS field is truncated, the field's attribute byte is inserted to adjust the truncation.

The following pages show examples of the scrolling behavior of mixed and DBCS fields.

Scrolling Mixed Fields

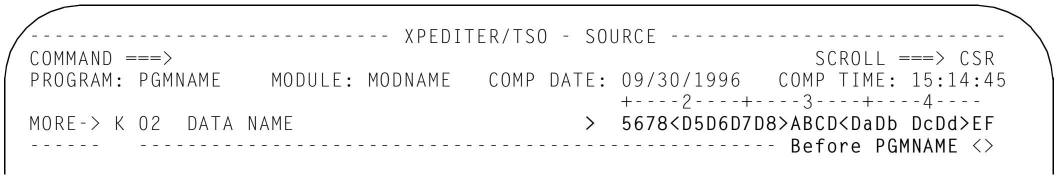
The following 45-byte string is used to show the truncation behavior in a mixed field:

```
----+----1----+----2----+----3----+----4----+
1234<D1D2D3D4>5678<D5D6D7D8>ABCD<DaDbDcDd>EFG
```

Example 1

When a SBCS character or SO/SI is at the left or right boundary of the window, no truncation occurs and the character is displayed.

For example, after scrolling the sample string 14 bytes left, the SBCS character 5 is at the left boundary of the Keep window and the SBCS character F is at the right boundary of the window. The result is shown below:



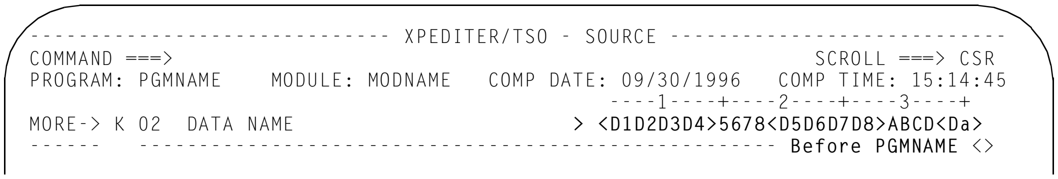
```
----------------------------- XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                     SCROLL ===> CSR
PROGRAM: PGMNAME    MODULE: MODNAME   COMP DATE: 09/30/1996   COMP TIME: 15:14:45
                                                  +----2----+----3----+----4----
MORE-> K 02  DATA NAME                         >  5678<D5D6D7D8>ABCD<DaDb DcDd>EF
------   --------------------------------------------------- Before PGMNAME <>
```

Figure G-1. SBCS Example 1—Scrolling The Sample String 14 Bytes Left

Example 2

When the first byte of a DBCS character is at the left boundary of the window, the character is displayed and an SO is inserted outside the window to maintain the SO/SI pairing.

For example, after scrolling the sample string 5 bytes left, the first byte of the DBCS character D1 is at the left boundary of the window. The result is shown below:



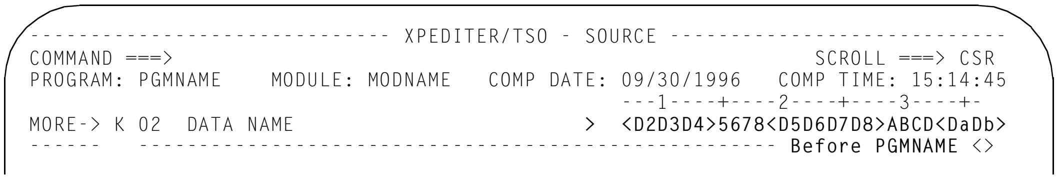
```
----------------------------- XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                    SCROLL ===> CSR
PROGRAM: PGMNAME    MODULE: MODNAME   COMP DATE: 09/30/1996   COMP TIME: 15:14:45
                                                 ----1----+----2----+----3----+
MORE-> K 02  DATA NAME                      > <D1D2D3D4>5678<D5D6D7D8>ABCD<Da>
------   ---------------------------------------------------- Before PGMNAME <>
```

Figure G-2. DBCS Example 2—Scrolling The Sample String 5 Bytes Left

Example 3

When the second byte of a DBCS character is at the left boundary of the window, the character is not displayed and an SO is inserted at the left boundary to maintain the SO/SI pairing. However, if you displayed the underlying hexadecimal values, the first byte of the window contains the hexadecimal value of the second byte of the truncated character.

For example, after scrolling the sample string 6 bytes left, the second byte of the DBCS character D1 is at the left boundary of the window. The result is shown below:



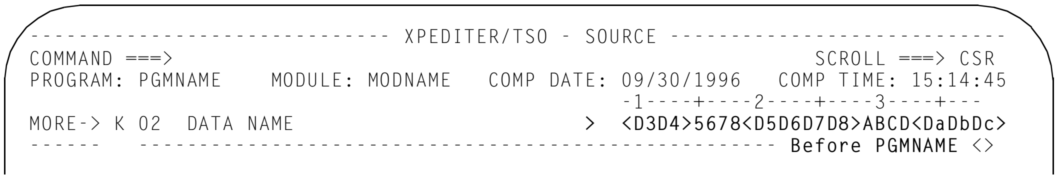
```
----------------------------- XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                    SCROLL ===> CSR
PROGRAM: PGMNAME    MODULE: MODNAME   COMP DATE: 09/30/1996   COMP TIME: 15:14:45
                                                 ---1----+----2----+----3----+-
MORE-> K 02  DATA NAME                       >  <D2D3D4>5678<D5D6D7D8>ABCD<DaDb>
------   ---------------------------------------------------- Before PGMNAME <>
```

Figure G-3. DBCS Example 3—Scrolling the Sample String 6 Bytes Left

Example 4

When the first byte of a DBCS character is at the right boundary of the window, the window is extended by one byte and the complete DBCS character is displayed. The first byte is displayed at the right boundary of the window and the second byte is displayed outside the window. An SI is also inserted outside the window to maintain the SO/SI pairing.

For example, after scrolling the sample string 5 bytes left, the first byte of the DBCS character Dc is at the right boundary of the window. The result is shown below:



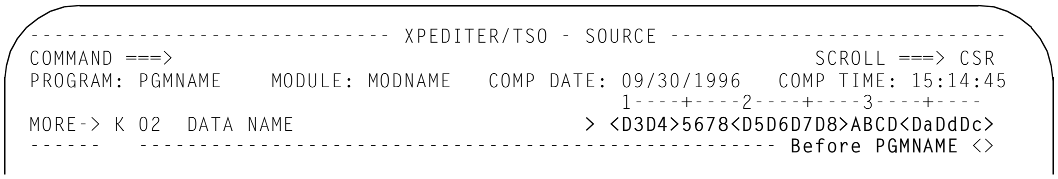
```
----------------------------- XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                    SCROLL ===> CSR
PROGRAM: PGMNAME    MODULE: MODNAME   COMP DATE: 09/30/1996   COMP TIME: 15:14:45
                                                 -1----+----2----+----3----+---
MORE-> K 02  DATA NAME                       >  <D3D4>5678<D5D6D7D8>ABCD<DaDbDc>
------   ---------------------------------------------------- Before PGMNAME <>
```

Figure G-4. DBCS Example 4—Scrolling the Sample String 5 Bytes Left

Example 5

When the second byte of a DBCS character is at the right boundary of the window, the character is displayed and an SI is inserted outside the window to maintain the SO/SI pairing.

For example, after scrolling the sample string 9 bytes left, the second byte of the DBCS character Dc is at the right boundary of the window. The result is shown below:



```
----------------------------- XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                    SCROLL ===> CSR
PROGRAM: PGMNAME    MODULE: MODNAME   COMP DATE: 09/30/1996   COMP TIME: 15:14:45
                                                 1----+----2----+----3----+----
MORE-> K 02  DATA NAME                       > <D3D4>5678<D5D6D7D8>ABCD<DaDdDc>
------   ---------------------------------------------------- Before PGMNAME <>
```

Figure G-5. DBCS Example 5—Scrolling The Sample String 9 Bytes Left

Scrolling DBCS Fields

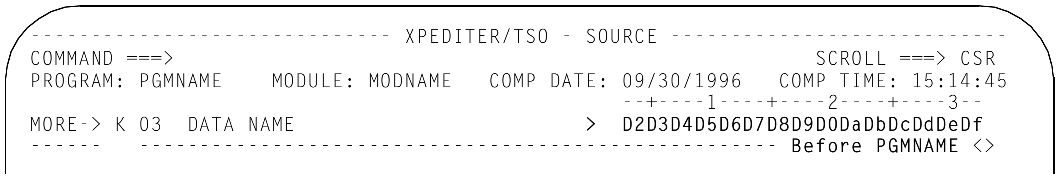
The truncation behavior is slightly different when scrolling DBCS fields. To show the behavior, the 40-byte DBCS string shown below is used:

```
----+----1----+----2----+----3----+----4
D1D2D3D4D5D6D7D8D9D0DaDbDcDdDeDfDgDhDiDj
```

Example 6

When the first byte of a DBCS character is at the left boundary of the window or the second byte of a DBCS character is at the right boundary, the character is displayed.

For example, after scrolling the sample string 2 bytes left, the first byte of the DBCS character D2 is at the left boundary of the window and the second byte of the DBCS character Df is at the right boundary of the window. The result is shown below:



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                     SCROLL ===> CSR
PROGRAM: PGMNAME    MODULE: MODNAME   COMP DATE: 09/30/1996   COMP TIME: 15:14:45
                                                 --+----1----+----2----+----3--
MORE-> K 03  DATA NAME                         > D2D3D4D5D6D7D8D9D0DaDbDcDdDeDf
------   ---------------------------------------------------- Before PGMNAME <>
```

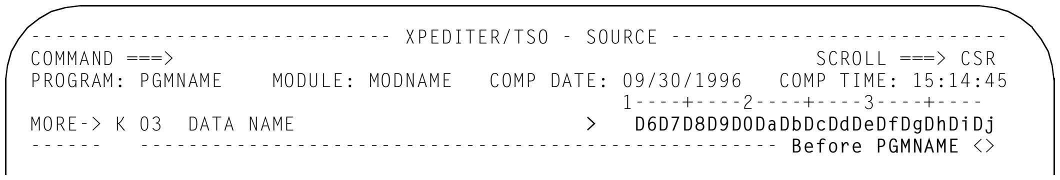
Figure G-6. DBCS Example 6—Scrolling The Sample String 2 Bytes Left

Example 7

When the second byte of a DBCS character is at the left boundary of the window, the character is not displayed and the attribute byte is inserted in the first byte of the window to adjust the truncation. Although the DBCS character is not displayed, the left boundary still contains the hexadecimal value of the second byte of the truncated character.

When the first byte of a DBCS character is at the right boundary of the window, the window is extended by one byte on the right and the character is displayed.

For example, after scrolling the sample string 9 bytes left, the second byte of the DBCS character D5 is at the left boundary of the window and the first byte of the DBCS character Dj is at the right boundary. The result is shown below:



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                     SCROLL ===> CSR
PROGRAM: PGMNAME    MODULE: MODNAME   COMP DATE: 09/30/1996   COMP TIME: 15:14:45
                                                 1----+----2----+----3----+----
MORE-> K 03  DATA NAME                         >  D6D7D8D9D0DaDbDcDdDeDfDgDhDiDj
------   ---------------------------------------------------- Before PGMNAME <>
```

Figure G-7. DBCS Example 7—Scrolling The Sample String 9 Bytes Left

Manipulating DBCS Data

DBCS data displayed in the Keep or Peek windows can be manipulated the same as SBCS data. However, depending on the field attribute (SBCS, DBCS, or mixed), there are some rules that must be observed when you type over, insert, delete, and move DBCS and SBCS data.

Manipulating DBCS data in the Keep and Peek windows follows the same truncation behavior as described in "Scrolling DBCS Data" on page G-3 and is controlled by the field attribute (SBCS, DBCS, or mixed). However, the underlying data beyond the visible window may be affected. Displaying the hexadecimal values with the PEEKH primary command or the PH or H line command lets you see the actual underlying values.

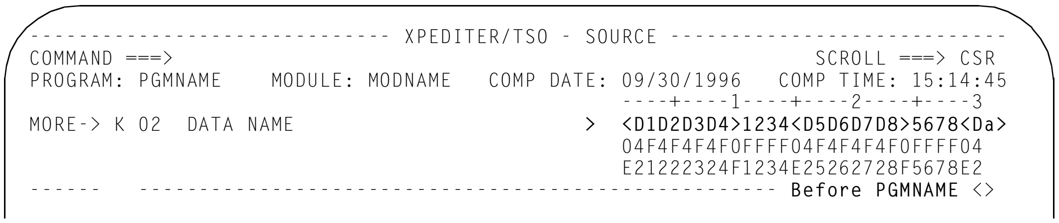
The following pages describe the behavior when you manipulate DBCS data.

Typing Over Data

Typing over a DBCS field with SBCS data is not allowed. You can type over a mixed field with SBCS or DBCS data. However, the underlying data beyond the visible window may be affected. The same applies when inserting and deleting data. For example, if you type over a DBCS substring with SBCS characters, the SO/SI bytes that are inserted or deleted are saved in the underlying data.

Example 8

The following mixed field is used in this example of typing SBCS data over DBCS data.



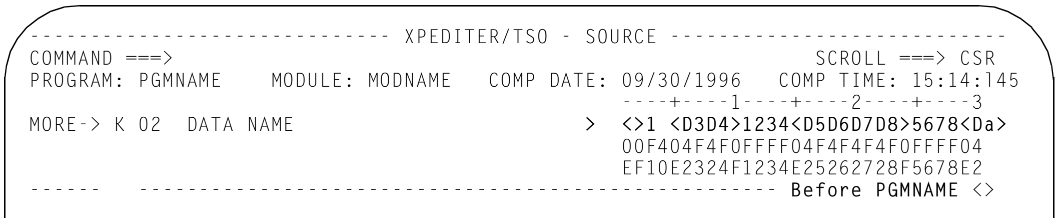
```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                      SCROLL ===> CSR
PROGRAM: PGMNAME    MODULE: MODNAME   COMP DATE: 09/30/1996   COMP TIME: 15:14:45
                                                 ----+----1----+----2----+----3
MORE-> K 02  DATA NAME                         > <D1D2D3D4>1234<D5D6D7D8>5678<Da>
                                                 04F4F4F4F0FFFF04F4F4F4F0FFFF04
                                                 E21222324F1234E25262728F5678E2
------   ---------------------------------------------------- Before PGMNAME <>
```

Figure G-8. Example 8—Typing SBCS Data Over DBCS Data

Type the SBCS character 1 over the first byte of the DBCS character D1.

Example 9

After pressing **Enter**, the window appears as follows:



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                      SCROLL ===> CSR
PROGRAM: PGMNAME    MODULE: MODNAME   COMP DATE: 09/30/1996   COMP TIME: 15:14:145
                                                 ----+----1----+----2----+----3
MORE-> K 02  DATA NAME                         > <>1 <D3D4>1234<D5D6D7D8>5678<Da>
                                                 00F404F4F0FFFF04F4F4F4F0FFFF04
                                                 EF10E2324F1234E25262728F5678E2
------   ---------------------------------------------------- Before PGMNAME <>
```

Figure G-9. Example 9—Typing SBCS Character 1 Over DBCS Character D1

Note that the hexadecimal values have changed. The following describes the result:

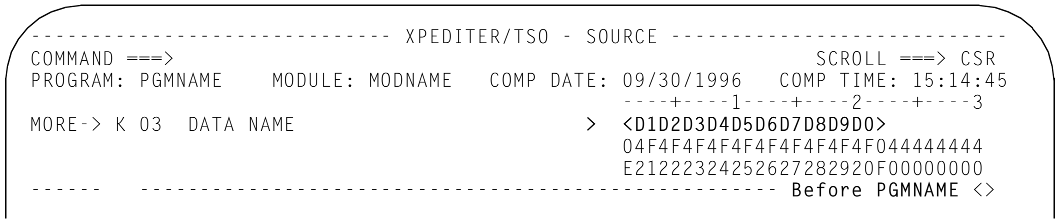
- The SO in byte 1 of the window is the original SO in the underlying data.
- The SI in byte 2 was inserted when you shifted to type in the SBCS character 1.
- The SBCS character 1 is displayed in byte 3. D1 does not appear because the first byte was typed over.
- A blank was inserted in byte 4 and a SO was inserted in byte 5 to show the beginning of DBCS data. D2 is not displayed because both bytes could not be displayed.
- Bytes 6, 7, 8, and 9 contain the DBCS characters D3 and D4.
- The SI in byte 10 is the original SI in the underlying data.

Inserting Data

Inserting characters is not allowed unless null characters (blanks) are present in the display window. If you try to insert more characters than the number of null characters, the terminal simply will not accept the extra characters.

Example 10

The following DBCS field is used for this example of inserting DBCS characters:



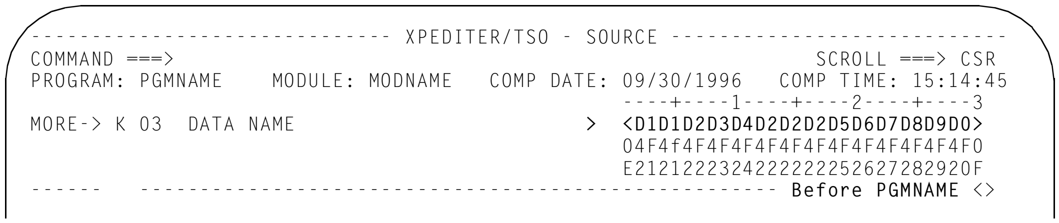
```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                    SCROLL ===> CSR
PROGRAM: PGMNAME    MODULE: MODNAME   COMP DATE: 09/30/1996   COMP TIME: 15:14:45
                                                 ----+----1----+----2----+----3
MORE-> K 03  DATA NAME                        >  <D1D2D3D4D5D6D7D8D9D0>
                                                 04F4F4F4F4F4F4F4F4F4F044444444
                                                 E21222324252627282920F00000000
------   ---------------------------------------------------- Before PGMNAME <>
```

Figure G-10. Example 10—Inserting DBCS Characters in a DBCS Field

Insert the DBCS character D1 in bytes 4 and 5 and repeat the character D2 in bytes 12 through 17.

Example 11

The result is shown below:



```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                    SCROLL ===> CSR
PROGRAM: PGMNAME    MODULE: MODNAME   COMP DATE: 09/30/1996   COMP TIME: 15:14:45
                                                 ----+----1----+----2----+----3
MORE-> K 03  DATA NAME                        >  <D1D1D2D3D4D2D2D2D5D6D7D8D9D0>
                                                 04F4f4F4F4F4F4F4F4F4F4F4F4F4F0
                                                 E2121222324222222252627282920F
------   ---------------------------------------------------- Before PGMNAME <>
```

Figure G-11. Example 11—Inserting DBCS Character D1 in Bytes 4 and 5

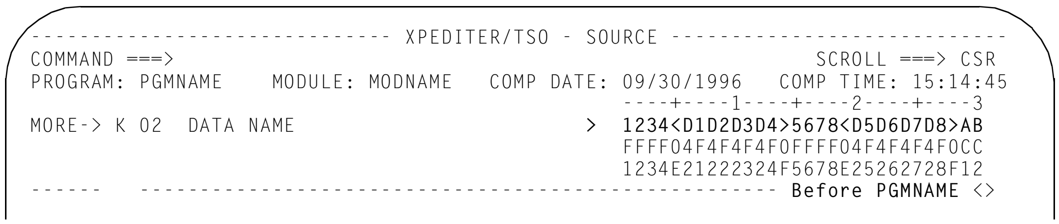
The result is that the inserted characters filled up the window. The The hexadecimal values show that the inserted characters are now a part of the underlying data.

Deleting Data

When characters are deleted with the DELETE key, the null characters (blanks) are filled from the end of the window, not from the end of the field. When characters are deleted with the ERASE (EOF) key, the window starting from the cursor position is blanked out, not the entire field.

Example 12

The following mixed field is used to show the behavior when deleting characters:



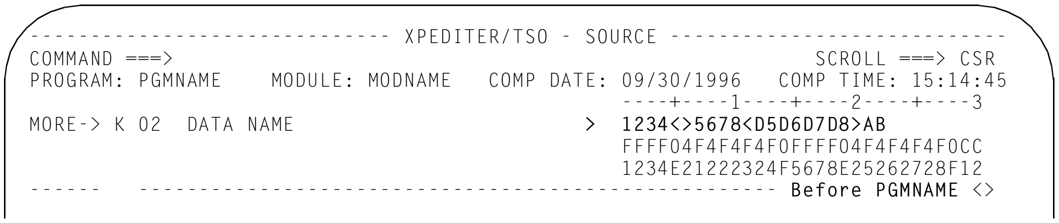
```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                    SCROLL ===> CSR
PROGRAM: PGMNAME    MODULE: MODNAME   COMP DATE: 09/30/1996   COMP TIME: 15:14:45
                                                 ----+----1----+----2----+----3
MORE-> K 02  DATA NAME                        >  1234<D1D2D3D4>5678<D5D6D7D8>AB
                                                 FFFF04F4F4F4F0FFFF04F4F4F4F0CC
                                                 1234E21222324F5678E25262728F12
------   ---------------------------------------------------- Before PGMNAME <>
```

Figure G-12. Example 12—Display of Behavior When Deleting Characters

Use the DELETE key to delete the DBCS characters D1, D2, D3, and D4. The following is displayed before you press the **Enter** key.

Example 13

Note that the hexadecimal values in the example below have not changed.



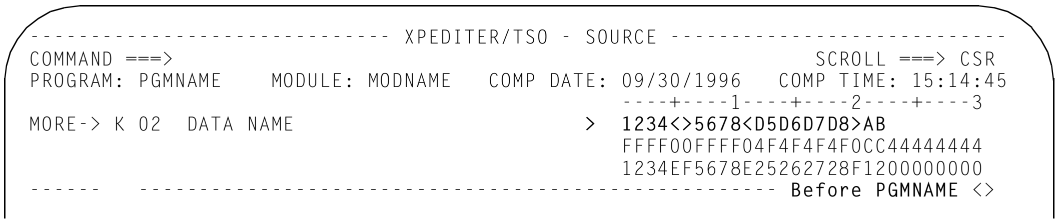
```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                    SCROLL ===> CSR
PROGRAM: PGMNAME    MODULE: MODNAME   COMP DATE: 09/30/1996   COMP TIME: 15:14:45
                                                 ----+----1----+----2----+----3
MORE-> K 02  DATA NAME                        >  1234<>5678<D5D6D7D8>AB
                                                 FFFF04F4F4F4F0FFFF04F4F4F4F0CC
                                                 1234E21222324F5678E25262728F12
------   ---------------------------------------------------- Before PGMNAME <>
```

Figure G-13. Example 13—Using the DELETE Key to Delete DBCS Characters

After pressing the **Enter** key, the following is displayed.

Example 14

Note that the hexadecimal values have changed—the values of the deleted characters are no longer displayed.



```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===>                                                    SCROLL ===> CSR
PROGRAM: PGMNAME    MODULE: MODNAME   COMP DATE: 09/30/1996   COMP TIME: 15:14:45
                                                 ----+----1----+----2----+----3
MORE-> K 02  DATA NAME                        >  1234<>5678<D5D6D7D8>AB
                                                 FFFF00FFFF04F4F4F4F0CC44444444
                                                 1234EF5678E25262728F1200000000
------   ---------------------------------------------------- Before PGMNAME <>
```

Figure G-14. Example 14—Deleted Characters No Longer Displayed

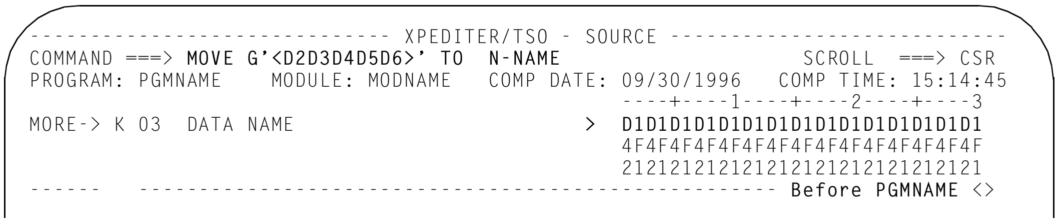
Moving Data

The MOVE command can be used to move DBCS or SBCS data into a mixed alphanumeric field. However, you cannot move a G type literal into this type of field.

A DBCS literal can be moved to a DBCS field as long as the appropriate SO/SI are inserted. When the DBCS data is moved to the field, the field is padded with the DBCS blank (X’4040’) and the SO/SI pairing is deleted.

Example 15

The following DBCS field is used to show the behavior when moving DBCS data:



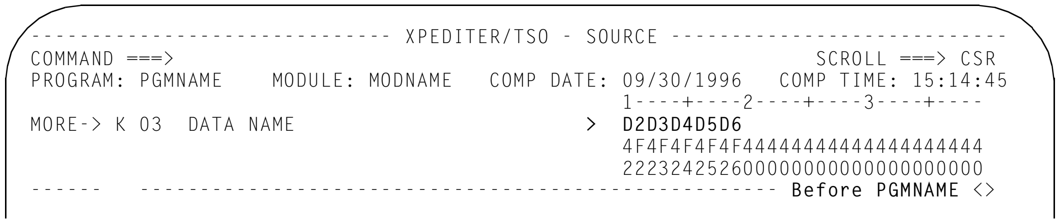
```
------------------------------ XPEDITER/TSO - SOURCE ---------------------------
COMMAND ===> MOVE G’<D2D3D4D5D6>’ TO  N-NAME                   SCROLL  ===> CSR
PROGRAM: PGMNAME    MODULE: MODNAME   COMP DATE: 09/30/1996   COMP TIME: 15:14:45
                                                 ----+----1----+----2----+----3
MORE-> K 03  DATA NAME                        >  D1D1D1D1D1D1D1D1D1D1D1D1D1D1D1
                                                 4F4F4F4F4F4F4F4F4F4F4F4F4F4F4F
                                                 212121212121212121212121212121
------   ---------------------------------------------------- Before PGMNAME <>
```

Figure G-15. Example 15—Displayed Behavior When Moving DBCS Data

Move the DBCS characters D2D3D4D5D6 to N-NAME.

Example 16

After pressing the **Enter** key, the move is performed and the window appears as follows: Note that the new string replaced the existing data and the field was padded with DBCS blanks.



```
------------------------------ XPEDITER/TSO - SOURCE ----------------------------
COMMAND ===>                                                        SCROLL ===> CSR
PROGRAM: PGMNAME    MODULE: MODNAME   COMP DATE: 09/30/1996   COMP TIME: 15:14:45
                                                 1----+----2----+----3----+----
MORE-> K 03  DATA NAME                         > D2D3D4D5D6
                                                 4F4F4F4F4F44444444444444444444
                                                 2223242526000000000000000000000
------   --------------------------------------------------- Before PGMNAME <>
```

Figure G-16. Example 16—Results of Moving DBCS Characters to N-NAME

Appendix H. COBOL-Structure Keywords

The following table describes each COBOL-structure keyword and the verbs that the keyword identifies. All verbs are shown in capital letters. Where keywords to a verb are listed, the keywords are shown in lowercase, and the verbs—capitalized—are enclosed in parentheses.

Keyword	Description and Reference
ALTer	Possible modification of the value of a data name or index: Everything in the **INPUT** keyword, plus ADD, by (PERFORM), CALL, COMPUTE, DIVIDE, EXAMINE (OS/VS COBOL only), from (PERFORM, WRITE), INITIALIZE (VS COBOL II only), INSPECT, MOVE, MULTIPLY, SEARCH, SET, STRING, SUBTRACT, TRANSFORM (OS/VS COBOL only), UNSTRING, varying (PERFORM); **EXEC CICS** keywords: ADDRESS, ASKTIME, ASSIGN, FORMATTIME, INQUIRE DATASET, INQUIRE PROGRAM, INQUIRE TERMINAL, INQUIRE TRANSACTION, LINK, XCTL

BRAnch	Transfer of logic control:
	
	Everything in the **CONDition** keyword, plus ALTER, CALL, CONTINUE, ENTRY, EXIT, GOBACK, GO TO, NEXT SENTENCE, PROCEDURE DIVISION, PERFORM, STOP, INPUT PROCEDURE, OUTPUT PROCEDURE;
	
	EXEC CICS keywords: ABEND, HANDLE ABEND, HANDLE AID, HANDLE CONDITION, LINK, RETURN, XCTL;
	
	EXEC SQL keywords: WHENEVER

CALL	CALL and CANCEL statements, excluding generated calls from EXEC CICS or EXEC DLI
CICS	EXEC CICS statements
CONDition	Conditional logic: At end (READ, SEARCH, RETURN), at eop/at end-of-page (WRITE), depending on (GO TO), else (IF), EVALUATE (VS COBOL II only), IF, invalid key (DELETE, WRITE, START, READ, REWRITE), otherwise (IF) (OS/VS COBOL only), ON (OS/VS COBOL only), on exception (CALL), on overflow (STRING, UNSTRING, CALL), on size error (ADD, SUBTRACT, MULTIPLY, DIVIDE, COMPUTE), times (PERFORM), until (PERFORM), when (SEARCH)
DLI	EXEC DLI statements or calls to CBLTDLI

Keyword	**Description and Reference**
IO	Input and output to a program: ACCEPT, CALL, CLOSE, DELETE, DISPLAY, EXHIBIT (OS/VS COBOL only), MERGE, OPEN, READ, RELEASE, RETURN, REWRITE, SORT, START, WRITE **Note:** The CALL modules must be specified on the Input/Output Module Definition screen completed at installation time or they will not be highlighted by the FIND IO command. **EXEC CICS** keywords: CONVERSE, DELETE, DELETEQ TD, DELETEQ TS, DBR, READ, READNEXT, READPREV, READQ TD, READQ TS, RECEIVE, RESETBR, RETRIEVE, REWRITE, SEND, STARTBR, NCPOINT, UNLOCK, WRITE, WRITEQ TD, WRITEQ TS; **CALL CBLTDLI function code** keywords: CHKP, CHNG, CLSE, CMD, DEQ, DLET, FLD, GCMD, GET CHN, CHNP, GHU, GN, GNP, GU, IRST, LOAD, LOG, OPEN POS, PURG, REPL, ROLB, ROLL, SHCD, SNAP, STAT, SYMCHKP, SYNC, TERM, XRST; **EXEC DLI** keywords: CHKP, DLET, GU, GN, GNP, ISRT, LOAD, LOG, REPL, ROLL, ROLB, SCHD, STAT, SYMCHKP, TERM, XRST; **EXEC SQL** keywords: CLOSE, COMMIT, DECLARE CURSOR, DELETE, FETCH, INSERT, OPEN, ROLLBACK, SELECT, UPDATE

Input or INPut	Receive data into the program:
	
	ACCEPT, READ, RETURN;
	
	EXEC CICS keywords: CONVERSE, READ, READNEXT, READPREV, READQ TD, READQ TS, RECEIVE, RETRIEVE;
	
	EXEC DLI keywords: GU, GN, GNP, STAT;
	
	CALL CBLTDLI function code keywords: CMD, FLD, GCMD, GET, GHN, GHNP, GHU, GN, GNP, GU, POS, STAT;
	
	EXEC SQL keywords: FETCH, SELECT, UPDATE

Output or OUTput	Transmit data out of the program:
	
	DELETE, DISPLAY, EXHIBIT (OS/VS COBOL only), MERGE, RELEASE, REWRITE, SORT, WRITE;
	
	EXEC CICS keywords: CONVERSE, DELETE, DELETEQ TD, DELETEQ TS, REWRITE, SEND, WRITE, WRITEQ TD, WRITEQ TS;
	
	CALL CBLTDLI function code keywords: CMD, DLET, FLD, ISRT, REPL, SNAP;
	
	EXEC DLI keywords: DLET, ISRT, REPL;
	
	EXEC SQL keywords: DELETE, INSERT, UPDATE

PARAgraph	Paragraph or section labels
SQL	DB2 statements or EXEC SQL calls

Glossary

abend. An error condition that results in abnormal termination of a program.

absolute address. An address that can reference the specified memory location without further modification.

active module. The module in which execution is suspended.

active source file. See current source display.

address. A unique nonnegative integer that identifies a byte location in main memory.

address space. The range of addresses available to a programmer.

alternate entry points. An alternate entry point is created whenever an ENTRY statement is used in COBOL, even if the ENTRY statement is the first statement in the Procedure Division.

array. A collection of one or more elements with the same characteristics grouped into one or more dimensions.

assemble. The preparation of a machine language program from a symbolic language program by substituting absolute operation codes for symbolic operation codes and absolute or relocatable addresses for symbolic addresses.

Assembler language processor. One of the language processors provided by Compuware, this language processor accepts Assembler output, builds work records, sorts and merges the records, and merges the records with the listing to produce processor control blocks that can then be used as input to other Compuware products.

base address. The beginning address of the storage area where the CSECT or DSECT resides.

base register. A general-purpose register used to store a base address.

batch. Processing in which jobs are grouped (batched). The jobs are executed sequentially, and each job must be processed to completion before the following job can begin execution.

BDAM. Basic direct access method.

Branch Coverage. When referencing XPEDITER/Code Coverage (if installed at your site), Branch Coverage is defined as a methodology for tracking the number of paths which were taken during execution of a conditional statement. For Code Coverage, this methodology designates a specific type of statistic. Branch Coverage is supported **only** for COBOL and **must** be implemented in order to indicate whether the next verb in succession is intended for execution. The collection of Branch Coverage statistics is optional during a Code Coverage session.

Qualifying condition statement types for Branch Coverage include IF, ON n, and WHEN. In terms of reliability, Branch Coverage is considered the most thorough method required for the testing of specialized application programs.

breakpoint. A location or offset (paragraph or statement) within the program where XPEDITER “breaks” or temporarily suspends normal program execution to perform another function.

C370 language processor. One of the language processors provided by Compuware, this language processor accepts C compiler output, builds work records, sorts and merges the records, and merges the records with the listing to produce processor control blocks that can then be used as input to other Compuware products

CA-OPTIMIZER. A COBOL productivity and optimization product for the OS and OS/VS environments. CA-OPTIMIZER optimizes COBOL programs. It works on the object code of a program, reducing the size of the program and its run time by eliminating redundant machine instructions.

case sensitivity. Whether a group of letters is uppercase or lowercase.

CICS. Customer information control system.

COBOL language processor. One of the language processors provided by Compuware, this language processor accepts COBOL compiler output, builds work records, sorts and merges the records, and merges the records with the listing to produce processor control blocks that can then be used as input to other Compuware products.

code coverage. The concept of measuring and reporting on how much of one or more programs have been executed by a set of tests. (see also XPEDITER/Code Coverage)

column template. A dashed line depicting column positions. When debugging with XPEDITER, a column template is displayed above each group item and each nonnumeric elementary item. The length of the column template corresponds to the length of the variable to be displayed as defined in the picture clause.

CMS. Conversational monitor system.

command area. The area of any display used to enter XPEDITER primary commands. The command area is designated as the left side of the second line of the source display.

command delimiter. A character used to separate commands in a list of commands entered simultaneously. The character used should be the same as the ISPF command delimiter. If the XPEDITER and ISPF command delimiters are not the same, ISPF will break the stream down into illogical sections to be passed on to XPEDITER one at a time for parsing.

command, CICS. In CICS, an instruction similar in format to a high-level programming language statement. CICS commands begin with the pseudo-verb EXEC (either EXEC CICS or EXEC DLI) and are terminated by END-EXEC. They can be issued by an application program to make use of CICS facilities.

command-language statement. Synonymous with command in relation to CICS.

command stacking. A method of entering multiple commands simultaneously by separating each command by a special delimiter. Command stacking is valid when testing with XPEDITER/TSO only if you are using ISPF version 2.1 or higher.

communication area (COMMAREA). An area that is used to pass data between tasks that communicate with a specific terminal. The area can also be used to pass data between programs within a task.

conditional expression. Any of the valid COBOL expressions supported by XPEDITER that test conditions to select between alternate paths of control depending upon the truth value of the condition.

CSECT. Control section.

current source display. The program currently displayed on the Source screen. It is named in the third line of the Source screen.

data area. Storage space defined and reserved at assembly time for insertion and manipulation of data at execution time.

date/time stamp. The date and time of compilation that marks the load module. XPEDITER stores the date/time stamp in the header record of the DDIO file. If there is a mismatch between the load module stamp and the DDIO file stamp, XPEDITER responds with a message in the log.

DB2 Stored Procedure. See Stored Procedure.

DDIO. A Compuware file access method.

DDIO file. A generic name for an XPEDITER source listing file.

declaration statement. In PL/I, a DECLARE statement that specifies the attributes of a name.

declarative. Directives that reserve defined areas of storage (DS statements) or define constant values (DC statements).

default delimiter. The delimiter set by XPEDITER.

dimension. The size of a table or array and the arrangement of its elements.

directive. A statement that tells the assembler to take a special action and generates no object code. For example, START, DSECT, and END are directives.

displacement. The number of bytes from the first byte of the storage area.

DL/I. Data language 1. IBM's database management facility provided by the IMS/VS database program products.

doubleword. A binary constant that has a length of eight bytes and can be aligned on a doubleword boundary (a location whose address is divisible by eight).

DSECT. Dummy control section. Used by Assembler to format an area of storage without producing any object code.

dump. Hexadecimal representation of storage that may contain data useful for diagnosing an error.

duplication factor. A value that indicates the number of times to generate the data specified immediately following the duplication factor.

effective address. The address that results from adding a base register value and a displacement value.

entry point. The alternate name supplied in the ENTRY statement on the link-edit control cards: By default in COBOL, it is the program name.

execution monitor. The XPEDITER execution processor is used to allocate the test data and environment to run the test, load and monitor execution of your program, and display and format the data in your program.

explicit declaration. In PL/I, a DECLARE statement that specifies the attributes of a name. Same as declaration statement.

figurative constant. A compiler-generated value referenced through the use of certain reserved words. The reserved word can be written in a program without having been defined.

File-AID for DB2. A DB2 database management and SQL development and analysis tool.

fullword. A binary constant that has a length of four bytes and can be aligned on a fullword boundary (a location whose address is divisible by four).

general-purpose registers. The 16 general-purpose registers are separate from main storage. They are numbered from 0 through 15 and are referenced by number. These 32-bit registers are used for binary arithmetic and to reference main storage positions by using base-displacement addressing.

halfword. A binary constant that has a length of two bytes and can be aligned on a halfword boundary (a location whose address is divisible by two).

HELP facility. Online support that can be invoked for clarification or aid in relation to a problem.

INCLUDE library. Under MVS, a partitioned dataset created and maintained by the user allocated to the ddname XINCLUDE. This library contains test scripts generated by a test session or used to set up a session.

INCLUDE test script. A predefined test script executed through the INCLUDE command. The commands in the test script are executed as they are read in, as if they had been entered serially from the terminal.

index register. A register whose content is added to (or subtracted from) the absolute address derived from a combination of a base address with a displacement.

initial test script. A special test script executed at the beginning of a session that is used to set up the testing environment. This test script is not executed through the INCLUDE command. Rather, it is specified on the appropriate environment test menu or command parameter.

intercommunication facilities. A term covering intersystem communication (ISC) and multiregion operation (MRO).

intersystem communication. Communications between separate systems by means of SNA facilities.

Keep window. The window at the top of the Source screen that automatically displays the values of data items referenced by the current execution line whenever execution halts. It also displays any data items specified by the KEEP command.

label. The entry in the name field of an Assembler language statement. The Assembler option supports label names of up to 30 characters.

language processor (LP). A processor that converts Assembler or compiler output into input for other Compuware products.

License Management System(LMS). Facility that enables you to centrally administer Compuware product License Certificates and manage access to Compuware products at your site. The LMS includes several components that enable you to establish, maintain, diagnose, and upgrade access to those Compuware products licensed by your enterprise. The LMS replaces the utility previously known as the Customer Profile Utility.

line commands. XPEDITER commands that are entered by typing over the compiler-generated statement numbers.

link pack area. In OS/VS2, an area of virtual storage containing reenterable routines loaded at IPL. It can be used concurrently by all tasks in the system.

literal. Any alphanumeric string of characters enclosed in apostrophes (' ').

load libraries. In the MVS environment, the set of partitioned datasets containing the link-edited application programs. XPEDITER searches the list for the load module of the program to be tested.

log. A file created and used by XPEDITER to record each command entered during a debugging session and the responses made to it.

macroinstruction. An instruction that causes the assembler to process a predefined set of statements called a macro definition. The statements from the macro definition replace the macroinstruction (or macro call) in the source program.

message area line. The line below the command area, generally the third line of any source display used to report brief error or informational messages.

MQSeries. MQSeries for OS/390 allows OS/390 applications to use ***message queuing*** in order to participate in message-driven processing. With message-driven processing, applications can communicate across different platforms by using the appropriate MQSeries products. All MQSeries products implement a common application programming interface for whatever platform the applications are able to run on. The calls made by the applications and the messages they exchange are common.

multiregion operation. Communication between CICS systems in the same processor without the use of SNA facilities.

MVS. Multiple virtual storage.

native CMS. The VM/CMS environment without the use of ISPF.

native TSO. The MVS/TSO environment without the use of ISPF.

nonrepresentable characters. Characters that cannot be printed or displayed. XPEDITER displays nonrepresentable characters as periods (.) by default.

NOQ environment. An XPEDITER/TSO environment used to test IMS MPP or BMP programs when BTS is not available for testing. IMS database calls can be made since
XPEDITER/TSO will provide addressability to any IOPCBs. However, all calls to the message queue must be skipped.

object module. A module that is the output of an Assembler or compiler.

offset. A relative location or position within a data area.

operating system. Software that controls the execution of jobs. It can provide resource allocation and scheduling.

PA keys. The terminal program access keys. Their definitions are unaltered by XPEDITER.

panels. Menus or screens presented on a display terminal.

paragraph. For COBOL, a paragraph is a subdivision of a COBOL program: A paragraph contains one or more statements or sentences that work as a unit to perform a specific set of operations.

During an Assembler test, XPEDITER/TSO considers a paragraph to be a name, a label on one or more valid executable instructions, or a name label on an EQU * or DS 0H that is followed by one or more valid executable instructions.

pausing. All pause breakpoint commands direct XPEDITER to unlock the keyboard and return control to the user. Any number of XPEDITER commands can be entered while XPEDITER is paused.

PCB. Program communication block.

PL/I language processor. One of the language processors provided by Compuware, this language processor accepts PL/I compiler output, builds work records, sorts and merges the records, and merges the records with the listing to produce processor control blocks that can then be used as input to other Compuware products.

primary commands. XPEDITER commands that are entered from the command area as opposed to XPEDITER line commands, which are entered by typing over the compiler-generated statement number.

procedure. In PL/I, a block of programming statements that starts from various points in a program by CALL statements and processes data passed to it from the *calling* block.

procedure division. The section of a COBOL program that contains executable instructions.

PSW. Program status word. An operating system control block defining the current status and location of a program that is executing.

Quickstart. A File Allocation Utility (FAU) enhancement which provides the ability for users to point XPEDITER at the JCL needed to run a test, therefore eliminating the need to use the FAU.

register. A device capable of storing a specified amount of data, such as one word.

review mode. An XPEDITER dynamic analysis feature for COBOL programs that lets the programmer review the execution flow backwards.

screens. Menus or panels presented on a display terminal.

script dataset/file. A dataset or file created and used by XPEDITER to record each executable command entered during a debugging session. The script dataset can be copied into an INCLUDE library, a sequential file, or another partitioned dataset to be used again.

scroll amount area. The furthest right area of the second line of the source display. It is used to display the current scroll amount, whenever scrolling is applicable.

scrollable fields. If a variable's length exceeds the screen width, the field becomes scrollable. Scrollable fields are identifiable by the highlighted MORE-> message in the line command area. When the screen is scrolled left or right, only the scrollable values and their associated column templates actually move.

scrolling. The ability to move the screen window across the data in any of four directions.

sentence. For COBOL, a sentence is a statement or group of statements that ends with a period.

session log. A file created and used by XPEDITER to record each command entered during a debugging session and the responses made to each command.

session script. A file created by XPEDITER containing the commands entered during a debugging session. This file can be saved as a member of an INCLUDE dataset to be used as input to another debugging session.

Shared Directory. A variable length record VSAM Relative Record Dataset (RRDS) that contains language processor (LP) directory records necessary to process LP database members.

Source display screen. The XPEDITER screen used to display the program source. Within this display screen, XPEDITER commands can be entered from the command area or by typing over the compiler-generated statement number. The data on the screen is scrollable in all four directions.

source listing. A compiled listing and other information about a file stored in a source listing file.

source listing file. A file containing source listings and accessed by DDIO.

split screen. A capability provided by ISPF/PDF that allows you to partition the display screen into multiple “logical” areas.

SQL. Structured query language.

Stored Procedure. A Stored Procedure is a user-written program that is stored at the DB2 server and can be invoked by a client application. A Stored Procedure can contain most statements that an application program usually contains. A Stored Procedure can also execute Structured Query Language (SQL) statements at the server as well as application logic for a specific function. A Stored Procedure can be written in COBOL, Assembler, C, PL/I or many other different languages, depending on the platform where the DB2 server is installed.

structure. A collection of data items.

SUB environment. The XPEDITER/TSO environment that is used to test stand-alone subroutines that make IMS database calls.

temporary breakpoint (GOTO). A breakpoint that occurs after a GOTO command is entered and before the target paragraph or statement is executed. The breakpoint is temporary and provides the opportunity to ensure that the target statement is the one intended.

test script. A predefined stream of XPEDITER commands used to set up, run, or rerun a debugging session.

test script library. A partitioned dataset allocated to the ddname XINCLUDE.

trace breakpoint. A breakpoint set by execution of the TRACE command in which modules are traced upon entry and exit.

unit testing subroutines. Testing a subroutine as a stand-alone program; that is, without the calling module being present.

user test scripts library. See INCLUDE library.

unattended batch. Processing data without interacting with the debugging session from your terminal. XPEDITER debugging commands are read from a test script and the output from the test session is written to the log.

VCON. Reserves storage for the address of a location in a CSECT that lies in another source module. A VCON is often used to branch to the specified external address.

VSAM. Virtual storage access method.

wide screen. A terminal screen that is 32 or 43 lines long, and 133 characters wide.

working storage. A section of a COBOL program used to define the data items that are used in a program.

Workload Manager (WLM). A component of MVS used for scheduling and dispatching of work within an MVS system. WLM is a prerequisite for the testing and debugging of DB2 Stored Procedures in an XPEDITER/TSO environment.

XPEDITER/Code Coverage. A product that collects run-time execution data from the XPEDITER/TSO, XPEDITER/CICS, and XPEDITER/IMS debuggers to help users analyze, understand, improve, and document how much of their code has been tested. Code Coverage interacts with XPEDITER/DevEnterprise to make code coverage results viewable online and to give users an idea of where the risk lies in their testing of a program.

XPEDITER ISPF interface. The XPEDITER ISPF interface is a collection of menus, CLISTs or EXECs, and programs created to assist you in allocating test data and supplying the information required to start an XPEDITER/TSO or XPEDITER/IMS debugging session.

XPEDITER/TSO. A symbolic debugging and testing tool used by COBOL, Assembler, PL/I, and C Language programmers to simplify the tasks of diagnosing and fixing programming errors by giving the programmer control over the execution of the program interactively at the source level.

Index