CA InterTest™ Batch

User Guide

Release 9.1.00

Third Edition

Documentation Changes

The following documentation updates have been made since the last release of this documentation:

- Screen shots of blank panels and unnecessary examples throughout the document were removed.
- Program names were updates throughout the document to reflect program names for CA InterTest Batch Release 9.1.00.
- Commands in keep window, variables using the DISPLAY and SET command are now case-sensitive.

CA Technologies Product References

This document references the following CA Technologies products:

- CA Endevor® Software Change Manager (CA Endevor SCM)
- CA File Master™ Plus
- CA InterTest™ Batch
- CA InterTest™ for CICS
- CA Librarian®
- CA Optimizer®
- CA Optimizer®/II
- CA Panvalet®
- CA Roscoe® Interactive Environment (CA Roscoe IE)
- CA SymDump® Batch

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 3: Core and the Map Options 49

Chapter 4: Debugging Commands 57

Appendix C: Advanced Demo Session 251

Chapter 1: System Overview

CA InterTest Batch is a full-screen, interactive, menu-driven testing and debugging facility for application programmers. It complements the CA InterTest for CICS product. The programmer controls the test session with the application commands.

This guide describes how to test a program using application batch commands and panels. The system is supported with HELP panels and an online tutorial, which describes the application options and commands.

You can use this product to test Assembler programs (assembled with Assembler F, Assembler H, or the High Level Assembler), COBOL programs (compiled with any COBOL compiler supported by IBM) or PL/I programs (compiled with any PL/I compiler supported by IBM).

This product grants the programmer complete access to the program as it is executing. You can stop the program and the Intercept panel displayed at user-specified intercept points. From the Intercept panel, you can browse the program listing in full-screen mode, using program function keys (PFKs) to scroll backward or forward through the program. You can examine and repair problem areas without ending the test session, or issue commands to set new intercept points and resume testing. Each application panel is supported with a HELP panel. An online tutorial describes the product system.

During a test session, you can do the following actions:

- Stop the test at any executable statement in the program.
- View the program listing while the test session is stopped.
- Trace the program's execution and browse the trace table entries.
- Display and alter data-item values.
- Step through the program in increments of any number of statements.
- Check the number of times each statement in the program is executed.
- Step through the program execution in reverse order.
- Access the other options such as Core.

The application runs under IBM's ISPF and CA Roscoe Interactive Environment (CA Roscoe) under ETSO. Therefore, programmers familiar with these environments need little or no training to use this product.

During a test session using the ISPF dialog manager, programmers can perform the following functions:

- Split the screen to access other ISPF options, such as EDIT.
- Print to the ISPF listing data set.
- Enter ISPF or TSO commands from any panel.

During a test session using CA Roscoe, programmers can perform the following functions:

- Split the screen to perform other CA InterTest Batch or CA Roscoe functions.
- Suspend the application session to perform other CA Roscoe functions.
- Print to an application data set, which can be allocated to the AWS or any MVS data set.
- Use the CA Roscoe print facility.

This section contains the following topics:

Operating Environment

Use your product to test any load module that runs under MVS, excluding those requiring authorization. COBOL and PL/I programs to be tested must be compiled with a compiler supported by IBM. Assembler programs must be assembled with the High Level Assembler.

The application ISPF requires the TSO environment with at least 1.5 megabytes of virtual storage specified in the SIZE operand of the LOGON command.

The application supports IBM's ISPF product release 2.1.0 and higher and CA Roscoe 6.0 and higher.

We recommend that the TSO command PROFILE WTPMSG be issued at logon. This command causes all messages that would normally appear in the job log of a batch job (write-to-programmer messages) to be sent to the terminal session.

Conventions

Panel Display Formats

The application uses full screen displays to prompt you for option selection and data entry, and to display source data, listings, and other information.

The first two lines are the header lines; they have a common format for all displays. The remainder of the screen contains a list of options, input fields and prompts, or scrollable data. The panel display format is shown next:

line 1 -->	Program Name	Title	Short Message
line 2 -->	Prompt or Input		Scroll Field

Line one of the application panel contains the panel title and short message area. The title line identifies the reason for the panel being displayed, and, where appropriate, data set information. The short message area displays the following information:

- Current line number (in BROWSE)
- Notice of successful completion of a processing function
- Notice of an error condition (accompanied by an audible alarm, if one is installed on the terminal)
- A frequency count in response to an Info line command

Line two, the prompt or input area, is used to enter an option, selection, or command.

You can scroll in any application panel containing a SCROLL field in line two. The application recognizes PAGE (P), HALF (H), MAX (M), or four numeric digits. The default is PAGE. Enter a numeric value in the SCROLL field to control the number of lines scrolled when a SCROLL key is pressed.

A scroll field value of cursor (CSR) is supported under CA InterTest Batch ISPF and CA Roscoe.

The application does not display ISPF long messages.

Program Function and Program Access Keys

For ISPF users, the application uses the PF keys as set from the ISPF option. For CA Roscoe users, the application uses the PF keys as set from the CA InterTest Batch Primary Option Menu.

Program Access Keys

The program access keys are defined as follows:

ATTENTION (PA1)	Interrupts the currently executing program and displays the ATTENTION INTERCEPT panel at the next executable statement in a monitored program. If the program is in a loop, you may need to press RESET before pressing PA1.
RESHOW (PA2)	Redisplays the contents of the screen. It may be useful if the CLEAR key was pressed accidentally, or if unwanted information has been typed but the Enter key (or a PF key) has not yet been pressed.

Note: The ATTENTION key can be either the PA1 key or the ATTN key, depending on the terminal type and how it is attached to the system.

Command Syntax Definition

The command syntax is described next. Valid long names and short names are shown in the examples of each command.

The following notation conventions are used to illustrate application commands:

- Uppercase letters and special characters are entered as shown.
- Lowercase italic letters are a generic description of a variable parameter's value.
- Brackets (< >) enclose optional values.
- Vertical bars (|) separate alternatives.
- Ellipses (...) specify that a parameter can be repeated.
- Default values are underlined.

Enter commands in uppercase or lowercase. Some commands can be abbreviated. The shortest abbreviation is the short name. If no short name is shown, you cannot abbreviate the command. You can abbreviate a long name a letter at a time down to the short name. All abbreviated versions of the command are valid.

Synonyms

Some commands have synonyms that you may find easier to remember, especially if you are a CA InterTest for CICS user. These synonyms can be used interchangeably. For example, you can enter the **AT** command, which sets an unconditional breakpoint, as **UNCOND**. As another example, you can enter **LDI**, which lists a data item in a keep window, as **KEEP**.

COBOL Statement Numbers

Some commands require a statement number in the command's syntax. For COBOL programs, these are the statement numbers that have been generated by the COBOL compiler. The compiler-generated statement numbers are the ones farthest to the left in the compilation listings and in application displays. These numbers must be used in all commands that require a statement number.

Assembler Offsets

When debugging an Assembler program, set breakpoints by using the statement number, as in COBOL programs, or by using the offset. When using an offset, the offset must be prefixed by the + sign.

For example, you may set a breakpoint at offset 78 by entering AT +78 in the Command area of the Intercept panel. Or use the Breakpoint panel and enter S in the Option field and +78 in the Statement field.

Debug Optimized Applications

This product supports the debugging of programs that have been optimized, either by the COBOL compiler's OPTIMIZE option or by
CA Optimizer or CA Optimizer/II. However, debugging these programs can sometimes result in unexpected behavior.

Note: The PL/I compilers are optimizing compilers.

Often as part of the optimization process, a compiler relocates individual instructions, statements, or even entire paragraphs so that the optimized program runs more efficiently. This means that some or all of the instructions generated for a given statement may be moved to another statement, or that some or all of the statements in a paragraph may be moved to another paragraph. When this type of optimization occurs, the resulting object program and corresponding listing may not accurately represent the relationship between the source statements and their generated object code, or even between a paragraph label and the statements contained within the paragraph. As a result, there may be times when the breakpoint intercept does not occur, or when the wrong sequence of statements appears to be executed while single-stepping. There may also be times when the debugger appears to highlight the wrong statement at a breakpoint intercept.

These unexpected displays do not indicate that a program is being executed incorrectly. They simply indicate that the debugger sometimes cannot accurately identify exactly which object code corresponds to which source statement, or which statement is contained within which paragraph.

The application uses the information in the compiler-generated procedure map or offset report to establish the program offset for each statement and label in the program. During execution, the debugger recognizes the start of the new statement or label by matching the program offset of the currently executing instruction with the PROTSYM information obtained from the compiler listing. Therefore, the accuracy with which the debugger can represent a breakpoint or other intercept is only as good as the information in the compiler listing.

Inaccuracies may include, but will not be limited to the following instances:

- Incorrect execution when using the SKIP, GO stmt# or CS stmt# commands
- Failure to stop at a breakpoint at a paragraph label or statement
- Unexpected or out of sequence highlighting of statements when single-stepping

Additionally, application abends may result from the use of the SKIP, GO stmt#, or CS stmt# commands because the optimized object code may have register requirements that do not support changes to the flow of control. These commands should be avoided when debugging an optimized program.

For the best debugging results, avoid using optimization whenever possible in your testing environment. Production applications may be compiled with optimization, and debugging these applications, as they exist without recompiling, is supported. However, be aware that you may experience some of the inaccuracies listed previously under these circumstances.

Primary Option Menu

The Primary Option Menu for ISPF users is displayed when you select the option on the ISPF/PDF Primary Option Menu.

You can access the following options from the Primary Option Menu:

1–Foreground

Test a program in foreground

2–Core

Display and alter virtual storage

3–Allocation

Allocations and JCL conversion

4–Map

Address space map

5–Batch

Batch link facility

U–Utilities

Symbolic utilities and preferences

X–Exit

Terminate the session

The Primary Option Menu for CA Roscoe users is displayed when you either select the CA InterTest Batch Option on the CA Roscoe menu or enter the following commands:

```
[xxx.]IBALLOC
[xxx.]IBRUN
```

where *xxx* is the prefix of the library where the application is installed. Omit this prefix if the application is installed in a common execution library.

You can access the following options from the Primary Option Menu for CA Roscoe:

1–Foreground

Test a program in foreground

2–Core

Display and alter virtual storage

4–Map

Address space map

5–Batch

Batch link facility

K–Keys

Display PF Keys

U–Utilities

Symbolic utilities and preferences

X–Exit

Terminate the session

System Structure

The following table shows the options. Availability of individual options depends on whether you are using ISPF or CA Roscoe. The application's primary function is option 1, Foreground. Other options are provided to support the test and debug facility, as shown in the following table:

Option	Name	Description
Option 1	Foreground	Lets you set up and control a test execution of a program.
Option 2	Core	Displays virtual memory in hexadecimal and character format. Use SCROLL keys and commands to locate and find data in virtual storage. **Note:** The application lets the programmer display/modify data with various commands. If it is necessary to place invalid data in a variable, use the Core Option.
Option 3	Allocation	Lets ISPF users allocate ddnames and allows conversion of JCL into a CLIST or ALIB.
Option 4	Map	Provides several address space-related displays that are most often used to debug complex system-related problems.

Option	Name	Description
Option 5	Batch	Lets you set up and test a program to be executed in Batch.
Option K	Keys	Lets CA Roscoe users set their PF keys for CA InterTest Batch. Type over the default assignments on the PF Key Definition panel and press Enter to change the PF keys. The default key assignments are as follows: PF1–Help PF2–Split PF3–End PF4–Return PF5–Find PF6–Go PF7–Up PF8–Down PF9–Swap PF10–Left PF11–Right PF12–Print Valid PF key commands are HELP, SPLIT, END, RETURN, FIND, GO, UP, DOWN, SWAP, LEFT, RIGHT, and PRINT.
Option U	Utilities	Provides the ability to display and update the contents of your PROTSYM file. Also lets you define which programs are to be excluded from auto-monitoring.
Option X	Exit	Terminates the application. The ISPF End key (normally PF3) or CA Roscoe End Key (normally PF4) also terminate the application from the Main Menu.
ISPF-HELP	Help	Displays the Tutorial Table of Contents from the Primary Option Menu. The tutorial summarizes the features and commands. You can page through the tutorial or select a subject from the Tutorial Table of Contents.

Chapter 2: Debugging Panels

Foreground, Option 1 on the Primary Option Menu, includes the following panels:

- Panels to identify the programs to be tested, and their associated symbolic files
- A panel to view the program listing and control the test execution
- Panels to display and alter program storage
- Panels to display trace, file, and other related information

Note: Novice users are encouraged to skim this chapter and the chapter "Debugging Commands" and read the Guidelines for Novice users in the chapter "Testing Procedures."

In addition, novice users can gain online experience with the application by performing the basic demonstration sessions in the appendices "Basic Foreground Demo Session" and "Basic Batch Link Demo" and by performing the advanced demonstration sessions in the appendix "Advanced Demo Session."

This section contains the following topics:

Overview of Test Panels

The testing process begins by selecting Option 1, Foreground, on the main menu.

The first two panels are called the Execution Control panel and the Monitor Control panel. Once you enter data on the Execution Control and Monitor Control panels, execution of the test program begins.

The Intercept panel acts as a window into the program where you can view any part of the program and from which you can issue control commands or line commands. The Intercept panel controls and monitors the test until the program comes to the normal end-of-job.

The Intercept panel displays the listing of the program, with the current statement highlighted. Enter commands to monitor the test step-by-step, to set breakpoints, to access other display panels, or to view other sections of the program.

Display the Intercept panel by any of the following conditions:

- Initial entry to the first monitored program
- A breakpoint in the program being reached
- Program ABEND
- Pressing the ATTENTION key
- STEP count being exceeded
- NEXT count being exceeded

The following table describes a list of test-related panels:

Panel	Description
BREAKPOINT CONTROL	Allows breakpoints (places where the application should interrupt) to be set by statement number or paragraph name.
BREAKPOINT DISPLAY	Lists the breakpoints set by statement numbers or paragraph names.
WHEN CONTROL	Allows breakpoints to be set based on variable contents or any time a variable changes.
WHEN CONDITION DISPLAY	Displays the when conditions set by either the line mode COND command or the When Control panel.
EQUATE	Allows data items to be equated to symbolic names. The variable may then be referred to using the data item or the symbolic name.
EQUATE DISPLAY	Displays the equate set by the line mode EQUATE command or the Equate panel
DATA DISPLAY	Displays a formatted display of program data.
LINKAGE DISPLAY	Displays a formatted display of the LINKAGE SECTION in COBOL format. This display is only available for COBOL programs.
RECORDS DISPLAY	Displays a formatted display of the records defined under the FILE SECTION in COBOL format. This display is only available for COBOL programs.

Panel	Description
FILE STATUS PANEL	Shows the allocation for each FILE DESCRIPTION by ddname and dsname, and the DCB information whether the file is open or closed. If it is open, it indicates for INPUT, OUTPUT or INOUT. This display is only available for COBOL programs.
PROGRAM TRACE DISPLAY	Displays program statements in the order of their execution.

Access the other options during the test session by using the split-screen feature and selecting another option from the Primary Option Menu.

Execution Control Panel

The first two panels (the Execution Control panel and the Monitor Control panel) specify the initial program to be executed and the names of the program to be tested.

Use the Execution Control panel to specify the main program, execution parameters, and allocations required to run the application. The data entered on this panel remains in place from session to session until the information is deleted or changed by you.

The Allocations Fields

The display shows the following fields:

ALIB DsnameMember

Use the ALIB Dsname and Member fields to specify which existing ALIB member to use to perform the necessary allocations, specify the parameters to be passed to the main program, or to identify the main program. If the ALIB contains multiple job steps, indicate which job step to use in the EXEC Job Step and Proc Step fields. Note that specifying an ALIB is not required.

Execution Overrides

Use the Execution Overrides fields to override any fields that were obtained through the ALIB, if one was specified. The fields are as follows:

PGM

Indicates and overrides the main program to be executed in this debug session. In the previous example, the main program name is obtained through the ALIB.

PARM

Specifies and overrides the execution parameter to be passed to the main program. In the previous example, the parameter in the ALIB member is overridden with FORMAT.

Note: PARM is mutually exclusive with the Number of Linkage Parameters field.

Number of Linkage Parameters

Specifies the number of linkage parameters expected by the main program when there is more than one. (For a COBOL program, this is equivalent to the number of Linkage Section data items defined on the PROCEDURE DIVISION USING statement.) You can initialize and update the parameter values using the SET and LINKAGE commands. It also allows you to debug a subprogram as a main program.

Task Libraries

Specify and override the STEPLIB (or JOBLIB if one was specified in the ALIB) required to execute the application. Specify the libraries by typing in the fully qualified data set name in quotes or by indicating a pre-allocated ddname in parenthesis.

Note: For customers with multiple CA Endevor SCM C1DEFLTS:

If you want to take advantage of the application dynamic symbolic support feature and you have chosen to run the debugging session as if you have a single C1DEFLTS (you have not defined your CA Endevor SCM site IDs in IN25SITE table, for example), you must now define the fully qualified data set name that contains the C1DEFLTS with your site ID in the Task Libraries fields.

Initial Commands

Specifies a member in your INT1CLIB DD that includes commands to be executed at the initial intercept.

Monitor Control Panel

Use the Monitor Control panel to specify which programs the application monitors during a debugging session. Explicitly define up to 30 programs, but the use of wildcards allows you to define an unlimited number of programs. The data you enter on this panel remains in place from session to session until you delete or change the information.

Monitored Programs

Use the Monitored Programs section to define the programs that you want to debug. The program name specified must match the name of the PROTSYM member that contains the symbolic information for the program being debugged. Entries may end in a wildcard, *.

Symbolic Files

Use the symbolic (PROTSYM) files section to define the files that contain the symbolic information for the programs that you want to debug. A PROTSYM entry must exist for each program that you want to debug. Specify at least one PROTSYM file in this section.

Dynamic Symbolic Support

Dynamic symbolic support, when activated, dynamically retrieves the compiler or assembler listing of the program being monitored from a CA Endevor SCM -managed listing data set and loads it into the designated PROTSYM file. For this feature to work, the load module library (specified under Task Libraries on the Execution Control panel) where the load module is loaded and the listing data set containing the module listing, must be under CA Endevor SCM control.

To activate the dynamic symbolic support feature, enter **Y** to the right of the PROTSYM file, underneath the heading Endevor Auto Populate to designate the PROTSYM file as the file to receive the listing. To deactivate the feature, change the **Y** to **N**.

The dynamic symbolic support feature, when activated, always loads the correct symbolic information whenever a matching symbolic version is not found.

Symbolic files associated with programs that do not contain a time stamp in the executable, such as non-LE-enabled Assembler programs, will be reloaded every time they are monitored. To suppress this behavior an additional option is provided on the Monitor Control panel (lower right hand corner under the heading: “Always Auto-Populate Non-LE-Enabled Assembler?”). Default is to always refresh the symbolic. This option when suppressed (by changing the **Y** to **N)** will never refresh the symbolic of a non-LE-enabled Assembler program.

Note: The dynamic symbolic support feature cannot differentiate between multiple listing outputs created by a single CA Endevor SCM processor for the same element. Thus, dynamic symbolic support using listing outputs from multiple compiles or assemblies from a single CA Endevor SCM GENERATE or MOVE action for the same element may produce unpredictable results.

Dynamic Symbolic Support Return Codes

Dynamic symbolic support provides an integrated service using API calls that deploy various proven components including CA Endevor SCM, PROTSYM post processors, z/OS dynamic allocation, CSVQUERY, and binder services. The following table provides a list of possible return codes (RC) and reason codes (RSC) the API and various components may return.

Note: Other numeric return codes and reason codes are also returned by CA Endevor SCM and z/OS binder services API (IEWBIND). See the appropriate manuals for details.

Function	RC	RSC	Meaning
API	0	0	Operations completed successfully.
API	2	0	GETMAIN failed.
API	8	0	CA Endevor SCM server not activated.
API	12	0	Invalid or missing parameter in NDVRCOMM.
API	16	0	Bad return code from CA Endevor SCM. See CA Endevor SCM documentation.
API	20	0	Data set open failed.
API	28	0	Allocation error. See IBAPILOG file.
API	32	0	IN25DALC load failed.
API	36	0	ENA$NDVR load failed.
API	40	0	CA Endevor SCM footprint not found for requested element.
API	44	0	Requested CSECT not found.
API	30	0	IN25CDRV load failed.
BINDER	44	0	Requested CSECT not found by binder.
CSVQ	48	0	CSECT from IDRU does not match CSECT in NDVR_NAM2.
Endevor	12	0	Incorrect C1DEFLTS used.
Endevor	32	15	CA Endevor SCM encountered critical error and cannot continue.
NDSB	4	0	Compressed footprint in IDRU invalid.
NIDR	2	0	GETMAIN failed.
NIDR	32	0	IN25NDSB load failed.
NSRV	8	NSRV	Listing server ABORTed or failed.

Function	RC	RSC	Meaning
0000	12	0000	Incorrect C1DEFLTS used.

PL/I Protsym/Load Module Map

If you entered a **Y** in the Monitor PL/I field on the Monitor Control panel, which implies you want to monitor the PL/I programs, and those programs were compiled with Visual Age PL/I for OS/390, you will see the PL/I Protsym/Load Module Map panel. In order to successfully monitor these programs, enter the load module in which these programs reside in the Modname field to the right of the entry. The following screen shows the PL/I Protsym/Load Module Map panel.

```
----------- CA InterTest Batch PL/I Protsym/Load Module Map ------------------



COMMAND ===>                                                 SCROLL ===> CUR

The following Visual Age PL/I programs are eligible for monitoring in this
execution. To monitor any of these programs, enter the name of the load module
in which the program resides. Press ENTER to continue or END to cancel testing.

 Protsym  Date     Time     Protsym DSN                              Modname
**************************** TOP OF DATA ***************************
 MYPLI    04/23/14 14:35:50 CAI.PROTSYM
 PLIDEMO  05/09/14 12:24:04 CAI.PROTSYM
 PLITEST  05/14/14 11:35:51 CAI.PROTSYM
**************************** BOTTOM OF DATA ************************
```

Access the Demo Session Options Menu

After setting the required breakpoints, you can access the Demo Session Options Menu by executing the demo program and choosing the Advanced Options Menu from the Welcome panel. Use the following steps:

Action: From the Initial Intercept panel, type **GO** and press Enter to execute the program.

Result: The demo program begins execution and displays the Welcome panel.

Action: Press **PF2** to access the Options Menu for the Advanced Demo Session.

Result: The Advanced Options Preliminary Screen displays, reminding you not to access the Options Menu without including the data set member DEMOINCL.

Action: Press Enter to continue.

Result: The Demo Session Options Menu displays.

Each of the Options on the menu is independent and can be performed in any order. To use an Option, follow the appropriate section. Upon completing an Option, you return to the Menu.

Before you start you should know that to interrupt a loop you must press the ATTN or PA1 key. Try the ATTN key first. If that does not interrupt the loop, your testing environment requires you to use the PA1 key. If using PA1, you may need to press RESET first.

Intercept Panel

The Intercept panel is the point where execution of the test program is controlled. The following screen shows the Breakpoint Intercept panel:

```
CAMRCOBB ---------- CA InterTest Batch *INITIAL* Intercept -------------------
COMMAND ===>                                                SCROLL ===> CUR
TRACE=> 000428
      000427        PROCEDURE DIVISION USING PARAMETER-AREA.
 *--> 000428        START-PROGRAM.
 *--> 000429            OPEN OUTPUT REPORT-OUT.
      000430       * ---------------------------------------------*
      000431       * Welcome to the CA InterTest Batch Link   *
      000432       * Demonstration Program. You are now at the      *
      000433       * Initial Breakpoint panel which is displayed    *
      000434       * upon entry to the first program to be tested.  *
      000435       * At this point, other breakpoints can be set,   *
      000436       * and control commands can be issued before      *
      000437       * the program is executed.                       *
      000438       * ---------------------------------------------*
 *--> 000439            MOVE  ZERO TO OPTION-CHOSEN.
 *--> 000440            MOVE  R1498-TIMEI TO R1498-TIME.
 *--> 000441            PERFORM TIME-EDIT THRU TIME-EDIT-EX.
      000442       *
 *--> 000443            MOVE  R1498-TIMEO TO R1498-TIME.
 *--> 000444            PERFORM TIME-EDIT THRU TIME-EDIT-EX.
      000445       * ---------------------------------------------*
      000446       * By default, this program will cause an S0C7    *
```

The Intercept panel has the following attributes:

- The monitor name of the program intercepted is displayed in the upper left corner of the panel (CAMRCOBB).
- The reason the program has been intercepted is displayed in the title line of the panel (*INITIAL* INTERCEPT).

- Application commands can be entered in the Command ===> field.
- The trace line shows the last ten statement numbers executed.
- The program listing displayed is scrolled to the current statement and that statement is highlighted.
- The display can be scrolled.
- The far left column may be used to enter line commands.

Note: Each time a test begins the application displays an Initial Intercept panel that stops before the first monitored executable statement is executed. At this point, use breakpoints and control commands before the program is executed.

The Intercept panel can optionally indicate the number of times each statement has been executed as shown in the following panel. This is called the *frequency counter*. For further information on the frequency counts, see the Counts Control Command in the chapter "Debugging Commands."

```
CAMRCOBB -------------- CA InterTest Batch BREAKPOINT INTERCEPT ---------------
COMMAND ===>                                                  SCROLL===> PAGE
TRACE=> 044500 043700 041500 038300 035600 033400 032100 025900 025800 025700
        044400
0001    044500      PERFORM 0110-FORMAT-DETAIL-LINE
        044600         VARYING VALUE-INDEX
        044700           FROM 1
        044800           BY 1
        044900      UNTIL VALUE-INDEX IS GREATER THAN 1.
        045000
--->    045100 9999-EXIT.
--->    045200      EXIT.
16                           16.48.01          OCT 26, 2014
0005    045400 0110-FORMAT-DETAIL-LINE SECTION.
0005    045500      SET COUNTER-INDEX TO INDEX-VALUE (VALUE-INDEX).
0005    045600      MOVE COUNTER-NAME(COUNTER-INDEX)         TO DETAIL-NAME.
0005    045700      MOVE COUNTER-NAME(COUNTER-INDEX)         TO DETAIL-ID.
0005    045800      MOVE COUNTER-READ(COUNTER-INDEX)         TO DETAIL-READ.
0005    045900      MOVE COUNTER-REJECTED(COUNTER-INDEX)     TO DETAIL-REJECTED.
0005    046000      MOVE COUNTER-USER-REJECT(COUNTER-INDEX)  TO DETAIL-USER-REJ.
0005    046100      MOVE COUNTER-DELETED(COUNTER-INDEX)      TO DETAIL-DELETED.
0005    046200      MOVE COUNTER-RECYCLED(COUNTER-INDEX)     TO DETAIL-RECYCLED.
0005    046300      DISPLAY DETAIL-LINE.
0005    046400 9999-EXIT.
```

Invoke all other test panels from the Intercept panel. You must return to the Intercept panel before accessing another test panel. For example, to format a data item, enter the DIsplay command in the Command field. To return to the Intercept panel, press the END key. To resume execution, enter **GO** in the Command field, use the G line command, or press the appropriate PF key. The test continues until the next intercept condition is encountered, an ABEND occurs, or the test comes to the normal end of job.

Types of Intercepts

The type of intercept being displayed is shown at the top center of the panel on the title line. The possible intercepts are defined as follows:

Intercept	Description
INITIAL INTERCEPT	Displayed upon entry into the first program that you have asked to monitor. To have the application stop upon entry to all programs, enter **AT ALL ENTRY** on the command line at this intercept.
UNCOND BEFORE and UNCOND AFTER INTERCEPT	Displayed when the program reaches an executable statement where the application was instructed by the programmer to stop executing the program.
ABEND INTERCEPT	Displayed when a monitored program ABENDs or an unhandled condition has been raised under LE/370. The ABEND code is shown in the title line. Resolve ABENDs by using application commands. For example, if invalid data is in a data item, use the set command to change the value. Continue the test using the GO command.
ATTENTION INTERCEPT	Displayed when a potential breakpoint is reached after the ATTENTION or PA1 key is pressed. For more information on use of the ATTENTION or PA1 key, see the Stopping a Looping Program in the chapter "Testing Procedures." Enter the GO command in the Command field or press the appropriate PF key.
STEP BEFORE and STEP AFTER INTERCEPT	Displayed when the count parameter in the STEP command is matched. Enter the STEP command in the Command field of the Intercept panel. For more information about this command format, see STEP Control Command in the chapter "Debugging Commands." The STEP command specifies the number of statements to be executed each time the GO command is issued. The STEP COUNT Intercept panel is displayed after each execution.
NEXT BEFORE and NEXT AFTER INTERCEPT	Displayed when the next count (set by the NEXT command) is exceeded. Enter the NEXT command in the Command field of the Intercept panel. For more information about this command format, see the NEXT Control Command in the chapter "Debugging Commands." The NEXT command causes the execution of the next *n* statements in the program. The NEXT command operand specifies the number of statements to be executed before the Next Count Intercept panel is displayed again.
WHEN when-name BEFORE and WHEN when-name AFTER INTERCEPT	Displayed at a conditional breakpoint. A conditional breakpoint is detected when the conditional statement specified in the COND command is true. Enter the COND command in the Command field of the Intercept panel to set conditional breakpoints or to access the When panel. For more information about this command format, see the WHEN (or COND) Control Command in the chapter “Debugging Commands.” Set conditional breakpoints with the line commands C, W, and V.

Intercept	Description
LABEL INTERCEPT	Displayed with executing a label when the AT ALL LABEL or AT LABEL breakpoint has been set.
DBCALL INTERCEPT	Displayed when a call is made to a database, such as DB2 or IMS, when the AT ALL DBCALL or AT DBCALL breakpoint has been set.
PGM ENTRY INTERCEPT	Displayed upon entry to any program that you are monitoring when the AT ALL ENTRY or AT ENTRY breakpoint has been set.
PGM EXIT INTERCEPT	Displayed before exiting any COBOL program that is being monitored when the AT ALL EXIT or AT EXIT breakpoint has been set.

Breakpoint Panels

There are two types of breakpoints, as follows:

- An unconditional breakpoint that is set by statement number or paragraph name.
- A conditional breakpoint that stops execution when a data item in the program contains (or does not contain) a particular value, or when the value of the data item changes (variable change breakpoint.)

Set conditional and unconditional breakpoints by entering a control command with operands on the Intercept panel, or by entering only the command and being prompted with a fill-in-the-blanks panel. This section describes how to use the fill-in-the-blanks panels to set breakpoints.

For information on setting breakpoints using control commands with operands, see the description of the following control commands in the chapter "Debugging Commands": AT (or UNCOND) and WHEN (or COND). For information on removing breakpoints, see the commands OFF (or OFFU) and OFFWN (or OFFC).

You can also set unconditional and conditional breakpoints by entering a single-character line command on the line where you want the breakpoint. For information on using line commands to set and remove a breakpoint, see the description of the U (or A), C (or W), V and X (or O) line commands in the Line Commands section of the chapter "Debugging Commands."

There is no limit to the number of breakpoints you can set.

Breakpoint Panel

The Breakpoint panel is a prompt panel used to set, reset (delete), or list unconditional breakpoints. An unconditional breakpoint stops program execution at the specified statement. Access the Breakpoint panel by entering the command UNCOND (or AT) or OFF, without operands, in the Command field of the Intercept panel.

Use the OPTION field to enter S (set), R (reset), or L (list):

S You must enter the statement number. The program stops whenever the statement number is encountered, unless you enter a count in the COUNT field.

The COUNT field is optional for the S option. Use it to specify the number of times a statement is executed before the program is stopped. The program stops at the designated statement prior to the final increment of the COUNT field. The commands to be executed execute when the COUNT limit is satisfied.

Use the AFTER field to indicate whether you would like execution to stop before this statement is executed (N) or after the statement is executed (Y).

R Deletes a breakpoint. You must enter the statement number. To delete a breakpoint for another program, enter the program name.

L Displays the Breakpoint Status panel. This is a listing of all unconditional breakpoints currently set in the program.

Note: You can also delete breakpoints from this screen.

Execute application commands at a breakpoint by entering the commands (separated by semicolons) in the COMMANDS TO BE EXECUTED AT BREAKPOINT field. The commands execute until the program resumes through the GO command, or until there is a request for a panel through a command such as DI or AT. The remaining commands are ignored. The list of commands can be up to 73 characters long and should be entered as shown in the following example:

```
COMMANDS TO BE EXECUTED AT BREAKPOINT:
===>SET BINARY-1 = 01;LDI BINARY-1;NEXT 5
```

In this example, the following commands are executed when the breakpoint occurs:

- The value of data item BINARY-1 is set to 1.
- Data item BINARY-1 is displayed on the Intercept panel in a *keep window*.
- Five statements are executed.

Panel Invocation

This panel is invoked by entering the UNCOND (or AT) or OFF command without operands on the command field from an Intercept panel.

Breakpoint Status Panel

Display the Breakpoint Status panel by selecting L (list) on the Breakpoint panel. It lists breakpoints set in the Breakpoint panel and by the UNCOND or AT commands for the current program. A sample Breakpoint Status panel follows.

```
------------------- CA InterTest Batch BREAKPOINT PANEL ----------------------
COMMAND ===>                                                 SCROLL ===> CUR

     *************************** Top of Data ********************************
     UNCONDITIONAL BREAKPOINTS FOLLOW:
     PROGRAM   STMT#   COUNT  EXEC  B/A  COMMANDS TO BE EXECUTED
     --------  ------  -----  ----  ---  -------------------------------------
.    CAMRCOBB  000589  0000   0000   B
     SOURCE LINE: SPLIT-SCREEN.

.    CAMRCOBB  000541  0000   0000   B
     SOURCE LINE: PAY-CALC.

.    CAMRCOBB  000659  0000   0000   B
     SOURCE LINE: ORDER-CALC.

.    CAMRCOBB  000561  0000   0000   B
     SOURCE LINE: CONDITIONAL-BREAKPOINT.

.    CAMRCOBB  000664  0000   0000   B
     SOURCE LINE: ORDER-CALC-EX.

.    CAMRCOBB  000714  0000   0000   B
```

The display shows the following items:

PROGRAM	This is the monitored program name.
STMT#	This is the statement number of the breakpoint.
COUNT	This shows the number of times a breakpoint is executed before the program stops and the command in the COMMANDS TO BE EXECUTED field is executed.
EXEC	This shows the number of times the breakpoint has been executed.
B/A	This indicates whether execution should stop before or after the statement.
COMMANDS TO BE EXECUTED	These are application commands specified in the Breakpoint panel or by the AT command.
SOURCE LINE	This shows the source line where the breakpoint was set.

Note: Use an O or X line command to remove these breakpoints from this screen.

You can use all SCROLL keys and SCROLL variables in this display.

When (or COND) Panel

Use the When panel to set, reset (delete), or list conditional breakpoints. It is invoked by entering either COND (or WHEN) or OFFWN, without operands, in the Command field of the Intercept panel.

You give each When condition a *when-name*. This name is up to eight characters long. If the when-name specified is a statement number in the program, the when condition is checked only before executing this statement. This line-specific when condition is known as a local conditional and requires less computing overhead since the condition is not tested before every statement.

A When-condition breakpoint is defined by one of two conditions:

- If only the when-name and the data-area-1 operands are specified, a W when-name Intercept panel is displayed each time the value of the data item specified in data-area-1 changes. This type of conditional breakpoint is called a variable change breakpoint.
- If the operands when-name, data-area-1, operator, and data-area-2 are specified, the data item specified in data-area-1 is compared to the data item specified in data-area-2 based on the operator. When the condition is true, a W when-name Intercept panel is displayed. (Data-area-2 can be a data item, a constant, LOW-VALUES, HIGH-VALUES, SPACES, or ZEROES.)

Use the OPTION field to enter S (set), R (reset), or L (list), as follows:

S The when-name and data-area-1 operands are required; the operator and data-area-2 operands are optional.

R Only the when-name operand is required.

L Displays the When Conditions Display panel. This is a listing of all conditional breakpoints in the program.

WHEN NAME A 1-8 character name assigned to the test or program statement number (required)

DATA-AREA-1 The data-name to be tested (required)

OPERATOR Condition which DATA-AREA-1 and DATA-AREA-2 must meet for the breakpoint to occur. Valid values are:

GT or > Greater Than

LT or < Less Than

EQ or = Equal

NE or ? = Not Equal

GE or >= Greater Than or Equal

LE or <= Less Than or Equal

Note: => and =< are not permitted.

DATA-AREA-2 (Optional field) Data-name or content which is compared with Data-AREA-1.

BEFORE/AFTER Field to indicate whether to stop before or after the statement when the condition is recognized.

Execute commands at a conditional breakpoint by entering the commands (separated by semicolons) in the COMMANDS TO BE EXECUTED AT WHEN CONDITION field. At the breakpoint, commands execute sequentially until the program resumes execution through the GO command, or until there is a request for a panel through a command such as the DI or UNCOND command. The remaining commands are ignored. The list of commands can be up to 73 characters long and should be entered as shown in the following example:

```
COMMANDS TO BE EXECUTED AT WHEN CONDITION:
===>SET BINARY-1 = 01;LDI BINARY-1;NEXT 5
```

Panel Invocation

This panel can be invoked by entering the COND (or WHEN) or OFFWN command without operands from an Intercept panel.

When Conditions Display Panel

Display the When Conditions Display panel by selecting L (list) on the When panel. It lists conditional breakpoints set in the When panel, by the V, C and W line commands, and by the COND (or WHEN) command for the current program with operands in the Command field of an Intercept panel.

A sample When Conditions Display panel follows:

```
------------------- CA InterTest Batch BREAKPOINTS PANEL ----------------------
COMMAND ===>                                                 SCROLL ===> CUR

     *************************** Top of Data ********************************
     CONDITIONAL BREAKPOINTS FOLLOW:
     PROGRAM   WHN NAME  DATA NAME 1                    OP    DATA NAME 2
     --------  --------  -----------------------------  ----  ---------------
.    CAMRCOBB  LOOPCOND  LOOP-OUT                       GT    +5000.
     ..SUBCOMMANDS FOR ABOVE WHEN ==> SET LOOP-OUT=0

    ************************* Bottom of Data ******************************
```

The display is defined as follows:

PROGRAM The monitored program name.

WHN NAME	This column shows the when-name.
DATA NAME 1	This column shows the data name specified in data-area-1.
OP	This column shows the operator that compares data-area-1 to data-area-2. When ANY is shown, the breakpoint occurs if any change is made to data-name-1.
DATA NAME 2	This column shows the data item or constant stated in data-area-2. The number of bytes to check are shown for an ANY condition.
B/A	Indicates whether execution should stop before or after the statement.
SUBCOMMANDS	To be executed when the breakpoint occurs are listed on the following line.
SOURCE LINE	Shows the source line where the breakpoint was set, if the breakpoint was local to a line

Note: Use an O or X line command to remove the breakpoint from this screen.

You can use all SCROLL keys and SCROLL variables in this display.

Equate Panel

Use the Equate panel to set, drop (delete), or list equated variable names. A data item may be equated with a symbolic name and then may be referred to by either the symbolic name or the data name. There is no limit to the number of equates that can be set. Access the Equate panel by entering the command EQU or DROP, without operands, in the Command field of the Intercept panel.

Use the OPTION field to enter S (set), D (drop), or L (list):

S You must enter an equate name and the data-name-1 fields. You need only enter the data-name-2 and data-name-3 fields when equating qualified data-items. Data-name-1 may be a subscripted or indexed data item.

If the equate name and data-name-1 fields are filled in, the S option need not be specified. Upon pressing Enter, the equate is set.

D Drops an equate. You must enter the equate name.

L Displays the Equate Display panel. This is a listing of all equates that have been set using the EQUATE command.

Note: For more information on the use and function of this command, see the EQUATE Control Command in the chapter "Debugging Commands (see page 57)."

Panel Invocation

This panel is invoked by entering the EQU or DROP command without operands from an Intercept panel.

Equate Display Panel

The Equate Display panel is displayed when L (list) is selected on the Equate panel or the LISTEQ command is specified on the Intercept panel. It lists all equates currently set.

The following screen is an example of an Equate Display panel:

```
------------------- CA InterTest Batch EQUATE DISPLAY PANEL -------------------
COMMAND==>                                                  SCROLL ===> PAGE
EQUATE NAME -------------------- ACTUAL NAME --------------------------------
****************************************** TOP OF DATA ***********************
WT1                             WA-T1-1A(1)
                                  OF WA-TABLE-1
N1                              NAME-ONE
                                  OF EMP-NAME
                                  OF PAYROLL-REC
******************************** BOTTOM OF DATA ******************************
```

The display shows the following fields:

EQUATE NAME The symbolic name equated with the data item

ACTUAL NAME The data item. Subscripts and qualifications are also shown.

You can use all SCROLL keys and SCROLL variables in this display.

Data Display Panels

You view or alter data values in a monitored program using data display panels. Enter the appropriate command in the Command field of the Intercept panel to access each type of display. The following table describes the commands:

Command	Function
DISPLAY	Displays a data item defined to the monitored program or the WORKING STORAGE SECTION of a COBOL program.
LINKAGE	Displays the LINKAGE SECTION of a COBOL program. This command is not valid for assembler or PL/I.

Command	Function
RECORDS	Displays the FILE SECTION file descriptions of a COBOL program. This command is not valid for assembler or PL/I.

All of the data display panels use the same format, so only the Display panel is shown in detail.

Note: When the DATAMON command is in effect and the trace navigation commands (PREV, ADV, POINT) have been used since the last breakpoint, this panel does not allow altering of data values. In addition, the data values that are displayed correspond to the data values before the current trace entry was executed.

Display Panel

A sample Display panel follows:

```
-------------- CA InterTest Batch WORKING STORAGE DISPLAY -------------------
COMMAND ===>                                              SCROLL ===> PAGE
ADDRESS  TYPE  LINE # LV  NAME  / VALUE                0----.----1----.----2-
002288BC       003900 02 MISCELLANEOUS-DATA
002288BC NB-S  004000 03 SOME-BIG-NUMBER              -123456789.
002288C0 NB-S  004200 03 SOME-BIGGER-NUMBER            +1234567890.
002288C8 NB-S  004400 03 SOME-BIGGEST-NUMBER          +1234567890123456789.
002288D0 F     004600 03 FLOATING-POINT-0              +00000000.00000E+05
002288D4 F     004800 03 ABEND-0CF                    +00000000.00000E+05
002288D8 AN    005000 03 FIRST-TIME-SWITCH             .
002288D9 AN    005200 03 INPUT-EOF-SWITCH              .
002288DA AN    005400 03 SOSV0003                      SOSV0003
002288E2 AN    005500 03 MODULE-NAME                   B00G0057
002288EC NB    005600 03 RECORD-DEVIATION             00000.00
002288F0 NB-S  005800 03 RECORD-DEVIATION-PLUS        +12345.67
002288F4 NB-S  006000 03 RECORD-DEVIATION-MINUS       -76543.21
002288F8 NP-S  006200 03 RECORD-DEVIATION-PACKED      +00077.25
002288FC NP-S  006400 03 RECORD-DEVIATION-SCALED      +000007654000.
00228901 NP-S  006600 03 RECORD-PERCENT-DEVIATION     +000875
00228903 NP-S  006900 03 RECORD-PERCENT-DEVIATION-MINUS -.000875
00228905 AN    007200 03 CURRENT-SYSID                ....
00228909 AN    007300 03 ABEND-0C7-DATA
00228909 NP-S  007400 03 ABEND-0C7                    ?404.
0022890B ND-SL 007600 03 EXTERNAL-DECIMAL-LEAD        -12345.99
```

The heading lines for all the panels are defined as follows:

COMMAND This field accepts display commands only; for valid commands, see Display Commands in the chapter "Debugging Commands."

ADDRESS This column is the starting virtual storage address for the data item. Use this address to address the data in Core, Option 2 on the PRIMARY OPTION MENU.

TYPE This is defined as follows:

Alphabetic

AN	Alphanumeric
AN-V	Varying alphanumeric
ANE	Alphanumeric edited
BF	Binary floating point
BIT	Bit
BIT-V	Varying bit

Display (Sterling nonreport)

DE	Display edited (Sterling report)
DBC	Graphic or DBCS
DBC-V	Varying graphic

Floating point (COMP-1/COMP-2)

FD	Floating point display (external floating point)
HX	Hexadecimal
N	National (UTF-16)
NB	Numeric binary unsigned (COMP)
NB-S	Numeric binary signed
ND	Numeric decimal unsigned (external decimal)
ND-OL	Numeric display, overpunch sign leading
ND-OT	Numeric display, overpunch sign trailing
ND-SL	Numeric display, separate sign leading
ND-ST	Numeric display, separate sign trailing
NE	Numeric edited
NP	Numeric packed decimal unsigned (COMP-3)
NP-S	Numeric packed decimal signed
PTR	Pointer data item (usage pointer)
UI	Index data item (usage index)

Subscripted

LINE # This column shows the program line numbers where the data names are defined.

LV This is the level number generated by the compiler. Level numbers appear in the DMAP output of the COBOL compiler or the AGGREGATE output of the PL/I compiler.

NAME/VALUE This column displays the data-item names.

The value of the data item is displayed as it is defined in the monitored program. If the data in the VALUE field is more than 20 characters long, it is shown on multiple lines.

Alter data by typing over the value field. If an attempt is made to modify a data item with a value larger than the size of the data item, the application truncates it. Movement of new values to the data area is done according to how the item is defined. For example, if the variable is alphanumeric and is 2 bytes long, if you enter 12B, the value truncates to 12.

You can use all SCROLL keys and SCROLL variables.

For assembler programs, when an access register or 64-bit address is requested for display, an additional line shows the access register or “high half” value, as shown in the following panels:

```
 -------------- CA InterTest Batch ASSEMBLER DATA DISPLAY ---------------------
COMMAND ===>                                                SCROLL ===> CUR
ADDRESS  TYPE  LINE # LV  NAME  / VALUE                0----.----1----.----2--
*************************** Top of Data ************** ********************
High-Half ==> 00000001
00000000 HX    000000 02 R1                            0000000000000000
00000008 HX    000000     +0008                        0000000000000000
00000010 HX    000000     +0016                        00000003000B3958
……….
*************************** Bottom of Data ************ ********************
```

```
-------------- CA InterTest Batch ASSEMBLER DATA DISPLAY ---------------------
COMMAND ===>                                                SCROLL ===> CUR
ADDRESS  TYPE  LINE # LV  NAME  / VALUE                0----.----1----.----2--
*************************** Top of Data ************** ********************
ALET     ===> 00010004
00000000 HX    000000 02 R3?                           0000000000000000
00000008 HX    000000     +0008                        0000000000000000
00000010 HX    000000     +0016                        0000000000000000
……….
*************************** Bottom of Data ************ ********************
```

For more information on Assembler indirect addressing, see the section Control Commands in the chapter “Debugging Commands”.

Invoke the panel by entering the Display command, with or without operands, from an Intercept panel. For assembler and PL/I programs, operands are required.

Linkage and Records Panels

Display the Linkage panel after the first PROCEDURE DIVISION statement has been executed, otherwise the linkage addresses will not be resolved. This occurs only when the Intercept panel is displayed for the first time and the LINKAGE command is entered. You can avoid this by using the NEXT command the first time the Intercept panel is displayed to execute the first statement in the PROCEDURE DIVISION and resolve the linkage addresses.

You can only use the RECORDS command, which displays the File Section Display panel, to modify data if the file is open.

Panel Invocation

Invoke these panels by entering the LInkage or REcords commands with or without operands from an Intercept panel. These commands are only valid for COBOL programs.

File Status Panel

The File Status panel displays the characteristics of all assigned files, and shows whether they are open or closed. The following screen is an example of the File Status panel:

```
-------------------- CA InterTest Batch FILE STATUS PANEL --------------------
COMMAND ===>                                                 SCROLL ===> PAGE
FD-NAME----------------------DDNAME---DSNAME---------------------------------
---
INPUT-FILE                    SYSUT1  INTC.SOME.DATAS
OPEN  INPUT   DSORG=PS RECFM=FB       BLKSIZE=06000 LRECL=0080
DCB=00123456  DEB=006C941C
.   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
SYMBOL-FILE                SYMBOL     *** WARNING *** DD CARD NOT ALLOCATED
 CLOSED        DSORG=PS RECFM=.....     BLKSIZE=00800 LRECL=0080
  DCB=0018D42C
.   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
OUTPUT-FILE                SYSUT2     INTC.NOTA.PDS
 OPEN  OUTPUT  DSORG=PS RECFM=FB       BLKSIZE=06000 LRECL=0080
  DCB=0018DD40 DEB=006C941C  DECB=0018BD80
.   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
```

A warning is issued for a file if the DD CARD is not allocated. In this product you can do any of the following actions:

- Split the screen and enter TSO using the ALLOCATE command to allocate the required file.
- Enter the TSO ALLOCATE command on the command line of the current ISPF screen.

- Invoke another session and select Option 3 to allocate the file.
- Suspend the session and perform the allocation under CA Roscoe, for CA Roscoe users. Then resume the ETSO session to resume your CA InterTest Batch session.

Note: Batch Link users must use the DDALLOC command. For more information on this command, see Control Commands in the chapter "Debugging Commands".

The data event block (DEB) is only displayed for files that have been opened and is not valid for closed files. The data event control block (DECB) is only displayed for basic sequential access method (BSAM) files.

For OS/VS COBOL users, IMS databases do not appear on the status display.

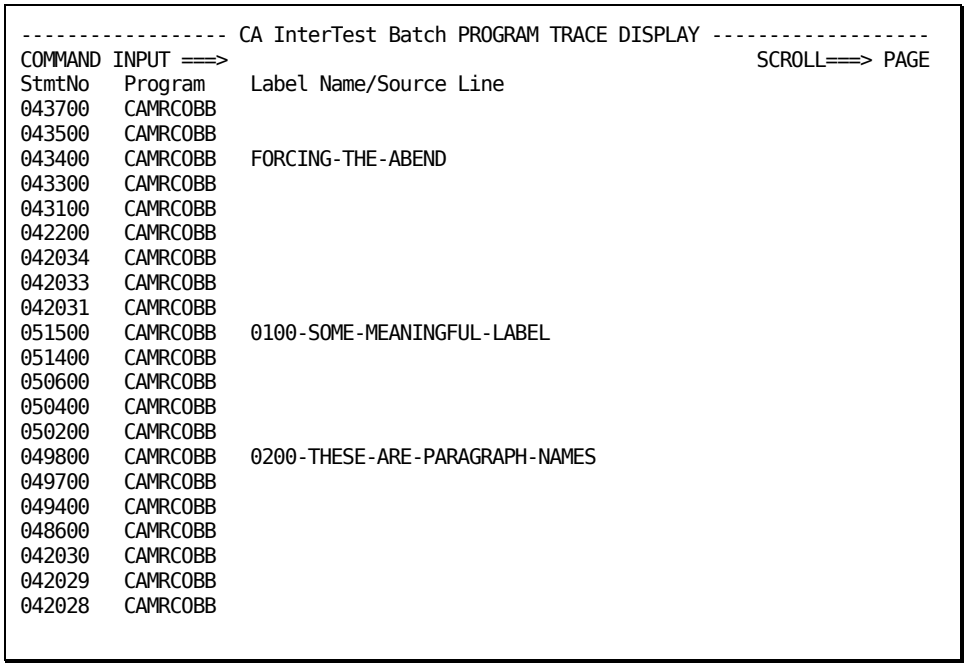
Panel Invocation

Invoke this panel by entering the FILES (FI) command from an Intercept panel. This command is only valid for COBOL programs.

Program Trace Display Panel

The Program Trace Display panel displays program statements in the order of their execution. To initiate a trace, enter the TRACE or TRACE SOURCE command in the Command field of the Intercept panel. For more details on the command format for TRACE Control Command, see the chapter "Debugging Commands".

A sample Program Trace Display panel with TRACE follows:



```
------------------ CA InterTest Batch PROGRAM TRACE DISPLAY -------------------
COMMAND INPUT ===>                                            SCROLL===> PAGE
StmtNo   Program    Label Name/Source Line
043700   CAMRCOBB
043500   CAMRCOBB
043400   CAMRCOBB   FORCING-THE-ABEND
043300   CAMRCOBB
043100   CAMRCOBB
042200   CAMRCOBB
042034   CAMRCOBB
042033   CAMRCOBB
042031   CAMRCOBB
051500   CAMRCOBB   0100-SOME-MEANINGFUL-LABEL
051400   CAMRCOBB
050600   CAMRCOBB
050400   CAMRCOBB
050200   CAMRCOBB
049800   CAMRCOBB   0200-THESE-ARE-PARAGRAPH-NAMES
049700   CAMRCOBB
049400   CAMRCOBB
048600   CAMRCOBB
042030   CAMRCOBB
042029   CAMRCOBB
042028   CAMRCOBB
```

A sample Program Trace Display panel with TRACE SOURCE follows:

```
------------ CA InterTest Batch PROGRAM TRACE DISPLAY -----------------------
COMMAND ===>                                                 Scroll ===> PAGE
StmtNo  Program   Label Name/Source Line
**************************************** TOP OF DATA ************************
001042  CAMRCOBB             MOVE  R1498-TIMEO TO R1498-TIME.
001041  CAMRCOBB          SPLIT-SCREEN.
001038  CAMRCOBB             IF   S-AID = 'PF3 ' OR 'PF15'
001037  CAMRCOBB             CALL 'ROETSAPI' USING  GETV  S-AID-DEF  S-AID.
001035  CAMRCOBB             IF RETURN-CODE NOT EQUAL ZERO
001034  CAMRCOBB             CALL 'ROETSAPI' USING  PANL  PANL-PARM.
001033  CAMRCOBB             MOVE '73' to  PANL-ID.
001028  CAMRCOBB          DEMONSTRATE-SPLIT-SCREEN.
000950  CAMRCOBB                 GO TO DEMONSTRATE-SPLIT-SCREEN.
000949  CAMRCOBB             IF  P-OPT = '3'
000947  CAMRCOBB             IF  P-OPT = '2'
000945  CAMRCOBB             IF  P-OPT = '1'
000942  CAMRCOBB             IF  S-AID = 'PF3 ' OR 'PF15'
000939  CAMRCOBB             IF RETURN-CODE NOT EQUAL ZERO
000937  CAMRCOBB             CALL 'ROETSAPI' USING  GETV S-AID-DEF   S-AID
000934  CAMRCOBB             IF RETURN-CODE NOT EQUAL ZERO
000933  CAMRCOBB             CALL 'ROETSAPI' USING  PANL  PANL-PARM.
000932  CAMRCOBB             MOVE '07' to  PANL-ID.
000930  CAMRCOBB             MOVE ZEROES    TO LOOP-OUT.
000929  CAMRCOBB          OPTIONS.
```

The display shows the following columns:

StmtNo	This is the statement number in the order of execution.
Program	This is the program ID for each statement number.
Label Name/ Source Line	This is the paragraph, procedure, or label name for the indicated statement number (TRACE) or the source code for the indicated statement number (TRACE SOURCE).

The Command field accepts only display commands; for valid commands, see Display Commands in the chapter "Debugging Commands." You can use all SCROLL keys and SCROLL variables in this panel.

Panel Invocation

Invoke the Program Trace Display panel by entering the TRACE command or the TRACE SOURCE command from an Intercept panel.

Chapter 3: Core and the Map Options

This chapter describes the valid commands in the DISPLAY-ALTER processor of Option 2 (Core) and the valid commands of Option 4 (Map).

This section contains the following topics:

Core Memory Display and Alter (Option 2: Core)

Core, Option 2 on the Primary Option Menu displays memory for your TSO user region. Use it during a test to display the program in memory. You can also enter core from any Intercept panel using the CORE command.

Memory is displayed in hexadecimal and character formats, and you can change memory that is not modify-protected by keying over the data where it appears on the display. Fetch protected memory cannot be displayed or modified.

Locate data in memory by using the scroll keys, the FIND commands, the DISPLAY command, or by keying in a specific address over the primary address field, as follows:

1. The FIND commands search for a specified character or hexadecimal string.
2. The DISPLAY command locates a specified address.
3. You can type an address over the primary address field, or an offset from the primary address can be typed over the primary offset field. The primary offset fields will be updated and the display scrolled to that part of memory.

The Display-Alter virtual memory panel is shown next:

```
----------------- CA InterTest Batch DISPLAY - ALTER MEMORY -------------------
COMMAND ===>                                              SCROLL ===> 00000100
HIGH-HALF ===> 00000000
ADDRESS   OFFSET     0--------4--------8--------C-----------0---4---8---C---
00006000  00000000 : 00000000 00000000 00000000 00000000  | ................ |
00006010  00000010 : 00000000 00000000 00000000 00000000  | ................ |
00006020  00000020 : 00000000 00000000 00000000 00000000  | ................ |
00006030  00000030 : 00000000 00000000 00000000 00000000  | ................ |
00006040  00000040 : 00000000 00000000 00000000 00000000  | ................ |
00006050  00000050 : 00000000 00000000 00000000 00000000  | ................ |
00006060  00000060 : 00000000 00000000 00000000 00000000  | ................ |
00006070  00000070 : 00000000 00000000 00000000 00000000  | ................ |
00006080  00000080 : 00000000 00000000 00000000 00000000  | ................ |
00006090  00000090 : 00000000 00000000 00000000 00000000  | ................ |
000060A0  000000A0 : 00000000 00000000 00000000 00000000  | ................ |
000060B0  000000B0 : 00000000 00000000 00000000 00000000  | ................ |
000060C0  000000C0 : 00000000 00000000 00000000 00000000  | ................ |
000060D0  000000D0 : 00000000 00000000 00000000 00000000  | ................ |
000060E0  000000E0 : 00000000 00000000 00000000 00000000  | ................ |
000060F0  000000F0 : 00000000 00000000 00000000 00000000  | ................ |
00006100  00000100 : 00000000 00000000 00000000 00000000  | ................ |
00006110  00000110 : 00000000 00000000 00000000 00000000  | ................ |
00006120  00000120 : 00000000 00000000 00000000 00000000  | ................ |
```

Note: If you enter the CORE command from an Intercept screen, the following panel displays:

```
----------------- CA InterTest Batch DISPLAY - ALTER MEMORY -------------------
COMMAND ===>                                              SCROLL ===> 00000100
EXPR ===> +00000000001B0E48      HIGH-HALF ===> 00000000 ALET ===> 00000000 ADDRESS
OFSSET     0--------4--------8--------C-----------0---4---8---C---
001B0E48  00000000 : 801B1000 00000000 00000000 00000000  | ................ |
001B0E58  00000010 : 00000000 00000000 00000000 00000000  | ................ |
001B0E68  00000020 : 00000000 00000000 00000000 00000000  | ................ |
001B0E78  00000030 : 00000000 00000000 00000000 00000000  | ................ |
001B0E88  00000040 : 00000000 00000000 00000000 00000000  | ................ |
001B0E98  00000050 : 00000000 00000000 00000000 00000000  | ................ |
001B0EA8  00000060 : 00000000 00000000 00000000 00000000  | ................ |
001B0EB8  00000070 : 00000000 00000000 00000000 00000000  | ................ |
001B0EC8  00000080 : 00000000 00000000 00000000 00000000  | ................ |
001B0ED8  00000090 : 00000000 00000000 00000000 00000000  | ................ |
001B0EE8  000000A0 : 00000000 00000000 00000000 00000000  | ................ |
001B0EF8  000000B0 : 00000000 00000000 00000000 00000000  | ................ |
001B0F08  000000C0 : 00000000 00000000 00000000 00000000  | ................ |
001B0F18  000000D0 : 00000000 00000000 00000000 00000000  | ................ |
001B0F28  000000E0 : 00000000 00000000 00000000 00000000  | ................ |
001B0F38  000000F0 : 00000000 00000000 00000000 00000000  | ................ |
001B0F48  00000100 : 00000000 00000000 00000000 00000000  | ................ |
001B0F58  00000110 : 00000000 00000000 00000000 00000000  | ................ |
001B0F68  00000120 : 00000000 00000000 00000000 00000000  | ................ |
```

The EXPR field is updateable. It contains a register expression or the address currently displaying. Commands that modify the displayed address also modify this field.

You can use all scroll keys in this display. Scroll amounts must be specified in hexadecimal. The scroll amounts apply to PF keys 7 and 8. PF keys 10 and 11 have different functions as follows:

PF Key	Description
PF10	Restore the address that was previously displayed.
PF11	Move the cursor to an address on the display and press the PF11 key. The address indicated by the cursor appears in the primary address field and the display is scrollable to that part of memory.

DISPLAY Command

The DISPLAY command sets the primary address field to the specified address. The primary address field is updated to the indicated address in virtual storage. The primary offset field is set to zero.

Syntax

Use the following syntax for DISPLAY:

`DISPLAY` *`address`*

Note: Data for variables are case-sensitive.

Variables

Use the following variables with the .label command:

Address

Displays the address in virtual storage. It may be a one to eight digit hexadecimal number.

FIND Command

The FIND command searches memory until the string operand is matched.

Syntax

Use the following syntax for FIND:

```
FIND string <NEXT|FIRST|LAST|PREV>
```

You can enter this command in its abbreviated form: F

Variables

Use the following variables with the .label command:

String	The search argument on the data area. String can be alphabetic, numeric, and special characters. If string contains imbedded blanks or quotes, you must enclose it in either single or double quotation marks.
NEXT	Causes a search for the next occurrence of string starting at the beginning of the first address displayed
FIRST	Causes a scroll to location zero followed by a FIND NEXT operation
LAST	Starts the scan at address 16,777,215 (that is, X'FFFFFF') and moves backwards to find the last occurrence of string
PREV	Starts the scan at the current address and moves backward to find the previous occurrence of the string

FX Command (Find Hex Data)

The FX command searches memory until the hexadecimal string is matched.

Syntax

Use the following syntax for FX:

```
FX string   <NEXT|FIRST|LAST|PREV>
```

Variables

Use the following variables with the FX command:

String	The search argument on the data area. String must be one to eight hexadecimal characters.

NEXT	Causes a search for the next occurrence of string starting at the beginning of the first address displayed. Next is the default.
FIRST	Causes a scroll to location zero followed by a FIND NEXT operation
LAST	Starts the scan at address X'FFFFFF' (that is, 16,777,215) and moves backwards to find the last occurrence of string
PREV	Starts the scan at the current address and moves backwards to find the previous occurrence of string

Usage Notes

In order to find the next occurrence of the string X'040C', enter the following command:

```
FX 040C
```

POP Command

The POP command restores from the stack the address most recently pushed onto the stack. The address becomes the primary address on the display. The primary offset field is set to zero.

Syntax

Use the following syntax for POP:

```
POP
```

Note: If the stack is empty, the following message appears:

EMPTY STACK

PUSH Command

The PUSH command places on the stack the current primary address.

Syntax

Use the following syntax for PUSH:

```
PUSH
```

Region Map Display (Option 4: Map)

Use the Map, Option 4 on the Primary Option Menu to display the task control blocks (TCBs) for your TSO session, including the program being tested. The commands to select the various control block displays are always shown at the top of the display area on the Region Map Display panel.

To display a list of all TCBs for your TSO session, use the command:

```
ALLTCB
```

Use your scroll keys to view and select the TCB number of the program you are testing from the list of ID numbers on the Control Block Display panel.

To set the TCB number for your program, enter the command ID with your task number. A sample Region Map Display follows:

```
-------------- CA InterTest Batch REGION MAP DISPLAY --------------------------
COMMAND ===>                                                 SCROLL ===> PAGE

   ID  - Set Current TCB    SAVE - Save area Map    TCB    - Task Status
   CDE - Pgms in Storage    ALLOC- Allocations      RB     - Active RB'S
                            FILE - Open Files       ALLTCB - All TCB's

   TCB: 007FE0A8            TIOT   : 00C71FE8       ASCB: 00F65580
   ID#: 00000001                                    ASID:     018C

--TCB -- -ID- ----------LEVEL------------- -PROGRAM- STATUS ---NDISP----
-------------------------------------------------------------------------------
007E5060 0009  . . . . . .8. . . . . ISPTASK   ACTIVE 040000000000
0079BBD8 0010  . . . . . .9. . . . . CALL      ACTIVE 040000000000
0079B908 0011  . . . . . .0. . . . . INTBATCH  ACTIVE 040000000000
007CD6E8 0012  . . . . . .1. . . . . GRANOLA   ACTIVE 040000000000
007CD550 0013  . . . . . .2. . . . . CAMRPSSC  ACTIVE 040000000000
007CD3B8 0014  . . . . . .3. . . . . CAMRPSWC  ACTIVE 040000000000
0079B160 0015  . . . . . .4XDMAP     ACTIVE 000000000000
0079B390 0016  . . . . . .4PCONTROL  ACTIVE 040000000000
0079A088 0017  . . . . . .1. . . . . IXCP      ACTIVE 040000000000
0079A3E0 0018  . . . . . .1. . . . . INTUTIL   ACTIVE 040000000000
007FFE88 0019  .1. . . . . . . . . . IEAVTSDT  ACTIVE 040000000000
------------------------------END OF TCB CHAIN--------------------------------
```

By using the appropriate commands, the Control Block Display panels display as shown in the following table:

Command	Control Block Display Contents
CDE	The programs in storage for the current TCB ID number
SAVE	The save area map for the current TCB ID number
ALLOC	The current allocations by ddname

Command	Control Block Display Contents
FILE	Information on open files related to the current TCB ID number
TCB	The status of the user's TCBs (not task related)
RB	The active request blocks (RB) for the current TCB ID number

The option ALLOC is similar to the TSO LISTALC command. Unlike the TSO LISTALC command, ALLOC displays the ddname. You can also scroll the display.

Note: For command FILE, no IMS database files appear on this display, because the application program does not own databases. IMS modules own them. They are displayed with the command ALLOC.

This panel supports valid scroll functions.

Chapter 4: Debugging Commands

This chapter describes the commands valid for use during a debugging session. There are four types of commands:

- Display
- Line
- Control
- Report

Use *display* commands to locate selected data in the program and position it on the screen. These are commands such as FIND and SCROLL; they can be used in any panel containing a command input field.

Control and *line* commands direct the test session and you enter them only in the Intercept panel. Line commands are shorter versions of some control commands.

Report commands generate reports to be printed and analyzed offline. Only enter report commands in the Intercept panel.

This section contains the following topics:

Display Commands

Display commands locate and position data on the screen. Enter them on the command line of any panel containing a command input field.

The following display commands are available to use with the application:

- COLUMN
- FIND
- FP (Find Paragraph)
- FS (Find Statement)
- KDOWN (Scroll Keep Window Down)
- KSIZE (Keep Window Size)

- KUP (Scroll Keep Window Up)
- LOCATE
- .label

COLUMN Command

The COLUMN command places a line that shows the column numbers on the first line of the display area of the Intercept or Display panel.

Syntax

Use the following syntax for COLUMN:

```
COLUMN <ON|OFF>
```

Variables

Use the following variables with the COLUMN command:

ON Enables the column numbers display.

OFF Disables the column numbers display.

If no argument is specified, ON is assumed.

COLUMN OFF is functionally equivalent to the RESET command, which removes the column numbers from the panel display.

FIND Command

The FIND command searches the file until the string operand is matched.

Syntax

Use the following syntax for FIND:

```
FIND string <col1 <col2>> <NEXT|FIRST|LAST|PREV>
```

You can enter FIND in its abbreviated form: F.

Variables

Use the following variables with the FIND command:

string	The search argument on the data area. String can be alphabetic, numeric, and special characters. If string contains embedded blanks or quotes, it must be enclosed in single or double quotation marks.
col1	Specifies the column in which to begin the search for string. If col1 is specified and col2 is omitted, the string must start in the specified column.
col2	Specifies the last column of a range of columns to search for string. If col2 is greater than the record length, the record length is used.
NEXT	Causes a search for the next occurrence of string starting at the beginning of the first line displayed after the cursor location. The default is NEXT.
FIRST	Causes a scroll to the top of the data followed by a FIND NEXT operation.
LAST	Starts the scan at the bottom of the data and moves backward to find the last occurrence of string.
PREV	Starts the scan at the current cursor location and moves backward to find the previous occurrence of string.

Usage Notes

To find the next occurrence of *TEST* in a program, enter:

```
F "'TEST'"
```

FP Command (Find Paragraph)

The Find Paragraph command scrolls the display to the definition of the specified paragraph, procedure, or label name.

Syntax

Use the following syntax for FP:

FP *paragraph-name*|*procedure-name*|*label-name*

You must enter this command in its abbreviated form: FP.

Variables

Use the following variable with the FP command:

paragraph-name	Designates a paragraph name defined in the Procedure Division of the currently qualified COBOL program.
procedure-name	Designates a procedure name defined in the currently qualified PL/I program.
label-name	Designates a label name defined in the currently qualified assembler or PL/I program.

FS Command (Find Statement)

The Find Statement command scrolls a display to a specified statement number.

Syntax

Use the following syntax for FS:

FS *statement-number*

You must enter this command in its abbreviated form: FS.

Variables

Use the following variables with the .label command:

statement-number	Specifies a statement number in the currently qualified program.

Cursor Sensitive Find

If a data name is displayed and you wish to locate its definition, place the cursor over the data name and press Enter. The application repositions the display at the definition of the data name.

If a paragraph, procedure, or label name is displayed and you wish to locate the line on which it starts, place the cursor over the name and press Enter. the application repositions the display at the paragraph, procedure, or label.

KDOWN Command (Scroll Keep Window Down)

Use the KDOWN command to scroll the keep window in the downward direction. The keep window appears on the Intercept panel immediately above the first line of the program listing.

Syntax

Use the following syntax for KDOWN:

```
KDOWN scroll-amt
```

Variables

Use the following variables with the KDOWN command:

scroll-amt	Identifies the number of lines to scroll, specified as one of the following:
PAGE	Scroll up one full page.
HALF	Scroll up one half page.
MAX	Scroll up to the top of the window.
nnn	Scroll the number of lines specified.

Usage Notes

The keep window is not scrollable if the number of entries in the window is less than the maximum depth defined by the KSIZE command. If the window is not scrollable, the KDOWN command is not functional.

If no argument is specified, the default scroll amount is used. The default scroll amount is defined in the upper right corner of the Intercept panel.

KSIZE (or KS) Command (Keep Window Size)

Use the KSIZE command to determine the size of your keep window. The keep window appears on the Intercept panel immediately above the first line of the program listing.

The KSIZE command lets you control the maximum depth of the keep window and whether the window should be fixed or variable in size.

Syntax

Use the following syntax for KSIZE:

```
KSIZE depth F|V
```

Enter this command in its abbreviated form: KS.

Variables

Use the following variables with the KSIZE command:

depth	The maximum number of lines of data to be displayed in the window at one time.
F	Specify F for a fixed window size.
V	Specify V for a variable sized window.

Usage Notes

You can set the keep window size to any depth from 1 to 15 lines of data. The default depth is 6 lines.

If KSIZE is defined as fixed, the keep window always occupies the maximum depth, even if the window contains less data items that the maximum. If the window is defined as variable, it will never occupy more lines on the screen than are necessary to display all of the data items, and never more than the maximum depth.

Define your keep window as fixed to prevent bouncing of your source listing while stepping through the program.

KUP Command (Scroll Keep Window Up)

Use the KUP command to scroll the keep window in the upward direction. The keep window appears on the Intercept panel immediately above the first line of the program listing.

Syntax

Use the following syntax for KUP:

```
KUP scroll-amt
```

Variables

Use the following variables with the KUP command:

scroll-amt	Identifies the number of lines to scroll, specified as one of the following:
PAGE	Scroll up one full page.
HALF	Scroll up one half page.
MAX	Scroll up to the top of the window.
nnn	Scroll the number of lines specified.

Usage Notes

The keep window is not scrollable if the number of entries in the window is less than the maximum depth defined by the KSIZE command. If the window is not scrollable, the KUP command is not functional.

If no argument is specified, the default scroll amount is used. The default scroll amount is defined in the upper right corner of the Intercept panel.

LOCATE Command

Use the LOCATE command to scroll to a specified line or a user-defined symbolic label in the program.

Syntax

Use the following syntax for LOCATE:

```
LOCATE nnnnnn|.label
```

You can enter this command in its abbreviated form: L.

Variables

Use the following variables with the LOCATE command:

nnnnnn	Specifies a relative line number. The display scrolls so that the line number is the first line of the display. A line number of zero causes a scroll to the top of the data; 999999 causes a scroll to the bottom of the data.
.label	Causes the display to scroll so that the line number equated to the symbolic label is the first line in the display.

A symbolic label equates to a line number. To establish a symbolic label, scroll the display until the line number to be equated is on the first line of the display. Enter the label in the COMMAND input field in the format:

xxxxxxx

The period is required and is followed by up to seven alphanumeric characters. The application does not retain symbolic labels when the test session ends.

.label Command

This command assigns a symbolic label to the current line number in the program.

Syntax

Use the following syntax for .label:

.*xxxxxx*

Variables

Use the following variables with the .label command:

xxxxxxx	The period is required and may be followed by up to 7 alphanumeric characters.

The application does not retain symbolic labels when the test session ends.

Line Commands

Line commands let you type a command in the far left field of a line. You may type multiple commands on different lines before pressing Enter, which causes them to be processed. Scrolling or any other command that causes the display to change also causes the specified line commands to be processed.

The following table is an overview of the Line commands, the corresponding Control Commands, and their function. The following pages provide a more complete description of each command.

Note that some commands have synonyms, such as **A** and **U**, which are fully interchangeable. Use whichever one you find easiest to remember.

Line Commands	Control Commands	Description
A U	AT UNCOND	An unconditional breakpoint is set at the specified line. A **U** remains in the first column to show where breakpoints are set. For assembler, if an A is placed next to a line in the keep window, the access register for that item is displayed.
C W	COND WHEN	A conditional breakpoint is set at the specified line. Issuing this command causes the application to display the When screen, where the condition may be specified. The application automatically completes the line number portion of the When screen based on your breakpoint line.
D	DISPLAY	The DISPLAY panel is shown for all of the data items on the line where the D command is specified. For assembler, the operands for the current instruction in register notation are shown in the DISPLAY panel. Cursor-sensitive DISPLAY - If **D** is entered and the cursor is placed on a COBOL or PL/I data item in that line, pressing Enter displays the DISPLAY panel for only that item and data items to the *right* of the item on the same line.
G	GO	The program begins executing (Go to) the line where the **G** is typed.
H	KEEPX LDIX LDX	The data items on the line where the **H** is placed are listed at the top of the Intercept panel in a keep window and shown in Hex format. For assembler, the operands for the current instruction in register notation are added to the keep window. Cursor-sensitive KEEPX - if **H** is entered and the cursor is placed on a COBOL or PL/I data item in that line, pressing Enter lists in the keep window only that data item and data items to the right of that item on the same line. For assembler programs, if an H is placed on a line within the keep window, the high half of the 64bit register is displayed.

Line Commands	Control Commands	Description
I	INFO	In instances where a frequency count exceeds 9999, Info displays the actual count at the top right of the screen. For assembler programs, if an **I** is placed on a line within the keep window, the access register is displayed.
K	KEEP LDI	The data items on the line where the **K** or **L** is placed are listed in a keep window on the top of the Intercept panel. For assembler, the operands for the current instruction in register notation are added to the keep window. Cursor-sensitive KEEP - If you enter **K** and the cursor is placed on a COBOL or PL/I data item in that line, pressing Enter lists in the keep window only that data item and data items to the right of that item on the same line.
O , blank, X	OFF OFFU	The breakpoint currently set on the line is reset (turned off).
P	PRINT	The DISPLAY for all the COBOL or PL/I data items on the line are written to the session log for later printing. For assembler, the operands for the current instruction in register notation are written to the session log for later printing.
R	REMOVE LDI	An RDI, reset data item, is done for each of the data items on the line to remove entries in the keep window. You may also use it next to individual keep window entries.
S	SKIP	Skip this statement. The **S** remains in the first column to show you which statements are skipped.
V	COND WHEN	A variable change breakpoint is set at the specified line. The breakpoint is triggered before the line is executed when the application determines that the value of a variable in the line changes. Cursor-sensitive variable change breakpoint - If you enter **V** and the cursor is placed on a COBOL or PL/I data item, pressing Enter designates that item as the variable change item.
X	OFF RDI	Remove breakpoint or remove data item from keep window.
+ nnn		Increment a subscript by *nnn* on a displayed data item.
- nnn		Decrement a subscript by *nnn* on a displayed data item.
<[nnnn]		Used on a keep window entry scrolls the data left *nnnn* bytes.
>[nnnn]		Used on a keep window entry scrolls the data right *nnnn* bytes.

A or U Command (Set Unconditional Breakpoints)

The A or U line command performs the same function as the AT or UNCOND command. An unconditional breakpoint is set on the line where the A or U is typed. The program stops at the statement prior to executing it. A U displays on the line to show you where breakpoints have been set. The U or A line command is only valid on executable lines of the program listing. Executable lines contain a frequency counter number or arrow. There is no limit other than the size of the screen. If you press Enter or the scroll keys, it sets the breakpoints.

You can type other commands over the U. Reset breakpoints using the OFF, O, X, or blank line commands.

The following panel shows breakpoints being set on lines 13100, 13400, and 13736 of the program:

```
CAMRCOB2 ---------- CA InterTest Batch NEXT BEFORE Intercept -----------------
COMMAND ===>                                                 SCROLL===> PAGE
TRACE= 012400 012300
         7        CAMRCOB2        15.49.31        JUN 30,2014
 0001        012300 PROCEDURE DIVISION USING PARMS TEST-THREE TEST-FOUR PASS-
  --->       012400 DEMO-BEGIN.
  --->       012500     GO TO DEMO-INIT.
  --->       012600 DEMO-INIT.
  --->       012700     OPEN OUTPUT OUTPUT01.
  --->       012800     MOVE HD1 TO LINE-OUT.
  --->       012900     PERFORM WRITE-LINE.

u--->        013100 FORCING-THE-ABEND.
  --->       013200     MOVE MASK1  TO BIN1.
  --->       013300     MOVE ZERO   TO MASK1.
u--->        013400     MOVE MASK1  TO BIN1
             013410                    BIN2
             013500                    BIN3
             013600                    BIN4.
  --->       013700 TABLE-MOVE.
u--->        013736       MOVE TABLE-FILL TO WA-TABLE-1.
  --->       013737 NUM-EDITED-TEST.
  --->       013738       MOVE IN-FUNC TO OUT-FUNC.
  --->       013739       MOVE IN-FUNC TO OUT-FUNC2.
```

Press Enter to set the breakpoints.

The listing appears as shown in the following panel. The lines that have breakpoints set display a U in the first column to show you where the breakpoints are.

```
CAMRCOB2 ---------- CA InterTest Batch NEXT BEFORE Intercept -----------------
COMMAND ===>                                                 SCROLL===> PAGE
 TRACE=> 012400 012300
         7        CAMRCOB2        15.49.31        JUN 30,2014
  0001        012300 PROCEDURE DIVISION USING PARMS TEST-THREE TEST-FOUR PASS-
   --->       012400 DEMO-BEGIN.
   --->       012500     GO TO DEMO-INIT.
   --->       012600 DEMO-INIT.
   --->       012700     OPEN OUTPUT OUTPUT01.
   --->       012800     MOVE HD1 TO LINE-OUT.
   --->       012900     PERFORM WRITE-LINE.

 U--->        013100 FORCING-THE-ABEND.
   --->       013200     MOVE MASK1  TO BIN1.

   --->       013300     MOVE ZERO   TO MASK1.
 U--->        013400     MOVE MASK1  TO BIN1
              013410                    BIN2
              013500                    BIN3
              013600                    BIN4.
   --->       013700 TABLE-MOVE.
 U--->        013736       MOVE TABLE-FILL TO WA-TABLE-1.
   --->       013737 NUM-EDITED-TEST.
   --->       013738       MOVE IN-FUNC TO OUT-FUNC.
   --->       013739       MOVE IN-FUNC TO OUT-FUNC2.
```

You may also use the LISTAT command to show the Breakpoints panel as follows:

```
-------------------- CA InterTest Batch BREAKPOINTS PANEL ---------------------
COMMAND ===>                                                 SCROLL ===> CUR

     *************************** Top of Data ********************************
     UNCONDITIONAL BREAKPOINTS FOLLOW:
     PROGRAM   STMT#   COUNT  EXEC  B/A  COMMANDS TO BE EXECUTED
     --------  ------  -----  ----  ---  ------------------------------------
.    CAMRCOB2  000589  0000   0000   B
     SOURCE LINE: SPLIT-SCREEN.

.    CAMRCOB2  000541  0000   0000   B
     SOURCE LINE: PAY-CALC.

.    CAMRCOB2  000659  0000   0000   B
     SOURCE LINE: ORDER-CALC.
 ********************************** BOTTOM OF DATA ************************
```

C or W Command (Set Conditional Breakpoint)

Use the C or W command to set a conditional breakpoint for a specified line. When you enter a C or W on a line and press Enter, the When screen displays. The WHEN-NAME field is populated with the line that you have specified. Use the When screen to specify the condition for the breakpoint. The condition is only checked before executing that statement. The application automatically completes the line number portion of the When screen based on your breakpoint line.

The C or W line command is only valid on executable lines in the program listing. Executable lines contain a frequency counter number or arrow.

Note: For further explanation of the When panel, see the chapter "Debugging Panels (see page 25)."

Example:

In the following example, you have entered the C command at line 1149 to set a conditional breakpoint.

```
CAMRCOB2 ---------- CA InterTest Batch NEXT BEFORE Intercept ------------------
Command ===>                                                  Scroll===> CUR
TRACE=> 001151 001150 001149 001148 001162 001155 001154 001153 001151 001150 0
U0001   001141               ORDER-CALC.
 0001   001142                   MOVE CUSTOMER-VALUES TO CUSTOMER-ORDER.
 0001   001143                   MOVE ZEROES TO ORDER-SUB-TOTAL, SUB-6, LOOP-O
 0001   001144                   PERFORM TOTAL-ORDER THRU TOTAL-ORDER-EX
        001145                       UNTIL SUB-6 EQUAL 5.
U--->   001146               ORDER-CALC-EX.
 --->   001147                   GO TO OPTIONS.
 0101   001148               TOTAL-ORDER.
c0101   001149                   ADD 1 TO SUB-6.
 0101   001150                   ADD 1 TO LOOP-OUT.
 0100   001151                   MULTIPLY UNIT-PRICE(SUB-6) BY NO-OF-WIDGETS(S
        001152                       GIVING ITEM-TOTAL.
 0100   001153                   ADD ITEM-TOTAL TO ORDER-SUB-TOTAL.
 0100   001154                   MOVE 1 TO SUB-6.
 0100   001155               TOTAL-ORDER-EX.
        001156              *-------------------------------------------------*
        001157              * The code above will loop. The loop can be      *
        001158              * interrupted by pressing an attention key.. .. *
        001159              * The frequency counter should provide a clue    *
```

When you press Enter in this example, the When panel displays, and the WHEN-NAME field is populated with the number 1149.

D Command (Display Working Storage)

For COBOL and PL/I programs the D line command causes the Display panel to be shown for all the data items on the line where the command is typed. For assembler programs, the D line command displays the operands for the current instruction in register notation. The D line command is valid on any line in the program listing.

You may type multiple D line commands on different lines, but to display the Display panel from each line in succession, press the END (PF3 or PF15) key. If D is specified on four lines, the Display panel appears for the data items on the first line. When END is pressed, the Display panel appears for the data items on the second line. This continues until the last D command is processed. Then the Intercept panel is displayed.

Cursor-sensitive Display: If you enter D and the cursor is placed on a COBOL and PL/I data item in the line, pressing Enter displays the Display panel for only that item and data items to the right of the item on the same line.

For more information on the Display Panel, see Data Display Panels in the chapter "Debugging Panels."

G Command (Go from Line)

The G line command causes the program to begin executing from the line on which you typed the G. Specify only one G line command at any time. If you type, execution begins from the highest line number. For example, if you type G on line 12900 and on line 13100, execution resumes at line 13100.

If you type G on line 13100 of the listing, as shown in the following panel, the program begins executing on line 13100 instead of normally beginning on line 12800. Lines 12800 and 12900 are not executed.

```
CAMRCOB2 ---------- CA InterTest Batch NEXT BEFORE Intercept -----------------
 COMMAND ===>                                                 SCROLL===> PAGE
 TRACE=> 012800 012700 012600 012400 012300
          7        CAMRCOB2        15.49.31        JUN 30,2014
  0001        012300 PROCEDURE DIVISION USING PARMS TEST-THREE TEST-FOUR PASS-
  0001        012400 DEMO-BEGIN.
   --->       012500     GO TO DEMO-INIT.
  0001        012600 DEMO-INIT.
  0001        012700     OPEN OUTPUT OUTPUT01.
 U--->        012800     MOVE HD1 TO LINE-OUT.
   --->       012900     PERFORM WRITE-LINE.

 g--->        013100 FORCING-THE-ABEND.
   --->       013200     MOVE MASK1  TO BIN1.
   --->       013300     MOVE ZERO   TO MASK1.
 U--->        013400     MOVE MASK1  TO BIN1
              013410                    BIN2
              013500                    BIN3
              013600                    BIN4.
   --->       013700 TABLE-MOVE.
 U--->        013736       MOVE TABLE-FILL TO WA-TABLE-1.
   --->       013737 NUM-EDITED-TEST.
```

H Command (Display Hex Data)

For COBOL or PL/I programs, the H line command causes a data item to be displayed in hexadecimal format for any data items contained on the line where you type the H. See the LDX command in the Control Commands section that follows for further explanation of the display. For assembler programs, the H line command displays the operands for the current instruction in register notation. For assembler programs, there is no functional difference between the H line command and the K line command. The number of H's that you specify is limited only by the number of data items displayed, because of the screen size.

Cursor-sensitive hex display: If you enter H and the cursor is placed on a COBOL or PL/I data item in that line, pressing Enter displays in hexadecimal format only that data item and data items to the right of that item on the same line.

You may reset the hex displays using the RDI command or the R line command.

Note: You cannot modify the zones and numerics portion of the data display. You also cannot modify group items.

In the following panel, the program has stopped executing because of an S0C7 abend as shown at the top of the Intercept panel. You have typed an H on line 13200 where the ABEND occurred to view the data items from that line. If you press Enter, it causes the line command to process.

```
CAMRCOB2 ---------- CA InterTest Batch ABEND S0C7 Intercept ------------------
COMMAND ===>                                                 SCROLL===> PAGE
 TRACE=> 013200 013100 020100 020000 012900 012800 012700 012600 012500 012400
         7        CAMRCOB2        15.49.31        JUN 30,2014
  0001         012300 PROCEDURE DIVISION USING PARMS TEST-THREE TEST-FOUR PASS-
  0001         012400 DEMO-BEGIN.
  0001         012500     GO TO DEMO-INIT.
  0001         012600 DEMO-INIT.
  0001         012700     OPEN OUTPUT OUTPUT01.
  0001         012800     MOVE HD1 TO LINE-OUT.
  0001         012900     PERFORM WRITE-LINE.

 U0001         013100 FORCING-THE-ABEND.
 h0001         013200     MOVE MASK1  TO BIN1.
  --->         013300     MOVE ZERO   TO MASK1.
 U--->         013400     MOVE MASK1  TO BIN1
               013410                    BIN2
               013500                    BIN3
               013600                    BIN4.
  --->         013700 TABLE-MOVE.
 U--->         013736       MOVE TABLE-FILL TO WA-TABLE-1.
  --->         013737 NUM-EDITED-TEST.
```

The following panel shows the hex data displays for the data items, MASK1 and BIN1, on line 13200. From the displayed data items, you should be able to determine the cause of the ABEND. A character field, MASK1, filled with blanks (X'40') is being moved to a numeric binary field. You could then use the SET command to change MASK1 to an acceptable numeric value.

```
CAMRCOB2 ---------- CA InterTest Batch ABEND S0C7 Intercept ------------------
COMMAND ===>                                                 SCROLL===> PAGE
TRACE=> 013200 013100 020100 020000 012900 012800 012700 012600 012500 012400
      ------------------------------------------------------------------------
      00243DA0 AN    004000 03 MASK1
               DATA OCCUPIES 0002 BYTES OF STORAGE     ZONES 44
                                                     NUMERICS 00

      00243DB2 NB-S  004600 03 BIN1                          +00000000.
               DATA OCCUPIES 0004 BYTES OF STORAGE     ZONES 0000
                                                     NUMERICS 0000
      ------------------------------------------------------------------------
         7        CAMRCOB2        15.49.31        JUN 30,2014
  0001         012300 PROCEDURE DIVISION USING PARMS TEST-THREE TEST-FOUR PASS-
  0001         012400 DEMO-BEGIN.
  0001         012500     GO TO DEMO-INIT.
  0001         012600 DEMO-INIT.
  0001         012700     OPEN OUTPUT OUTPUT01.
  0001         012800     MOVE HD1 TO LINE-OUT.
  0001         012900     PERFORM WRITE-LINE.

 U0001         013100 FORCING-THE-ABEND.

  0001         013200     MOVE MASK1  TO BIN1.
  --->         013300     MOVE ZERO   TO MASK1.
 U--->         013400     MOVE MASK1  TO BIN1
```

For assembler programs, if you use the H line command within the keep window, the high half of the 64bit register is displayed in a short message at the top of the screen. The following panel shows the short message displayed when the H command is used in the keep window:

```
MEMTEST  ---------- CA InterTest Batch *INITIAL* Inte- High Half = 00000001
COMMAND ===>                                                SCROLL ===> CSR
TRACE=> 000060 000059 000058 000056 000055 000050 000049 000048 000047 000046 0
      ------------------------------------------------------------------------
      00000000 HX    000000 02 R1                        0000000000000000
      ------------------------------------------------------------------------
 0001 000096 900F 1000            00000    59          STM   R0,R15,0(R1)
 ---> 00009A EB0F 1040 0026       00040    60          STMH  R0,R15,64(R1)
 ---> 0000A0 9B0F 1080            00080    61          STAM  R0,R15,128(R1)
```

I Command (Display Full Frequency Count Information)

In instances where a frequency count for a line exceeds 9999, the I line command displays the actual count at the top right of the screen. After typing I to the left of the line, press Enter to display the frequency count.

The following panel shows the actual frequency count for a line number that has exceeded 9999 executions:

```
CARMCOB2 ---------- CA InterTest Batch WHEN LOOPCOND BEFORE Inte Executed 10000
Command ===>                                                  Scroll===> CUR
TRACE=> 001151 001150 001149 001148 001162 001155 001154 001153 001151 001150 0
 U0001   001141               ORDER-CALC.
  0001   001142                   MOVE CUSTOMER-VALUES TO CUSTOMER-ORDER.
  0001   001143                   MOVE ZEROES TO ORDER-SUB-TOTAL, SUB-6, LOOP-O
  0001   001144                   PERFORM TOTAL-ORDER THRU TOTAL-ORDER-EX
         001145                       UNTIL SUB-6 EQUAL 5.
 U--->   001146               ORDER-CALC-EX.
  --->   001147                   GO TO OPTIONS.
  0101   001148               TOTAL-ORDER.
 I010K   001149                   ADD 1 TO SUB-6.
  0101   001150                   ADD 1 TO LOOP-OUT.
  0100   001151                   MULTIPLY UNIT-PRICE(SUB-6) BY NO-OF-WIDGETS(S
         001152                       GIVING ITEM-TOTAL.
  0100   001153                   ADD ITEM-TOTAL TO ORDER-SUB-TOTAL.
  0100   001154                   MOVE 1 TO SUB-6.
       PP 5668-958 IBM VS COBOL II Release 3.2 09/05/90               COB2DEMR
```

For assembler, if you use the I line command within the keep window the access register of the item is displayed in a short message at the top of the screen. The following panel shows the short message displayed when you use the A command in the keep window.

```
TESTAR   ---------- CA InterTest Batch ABEND S0C7 Int- ALET ===>  00010004
COMMAND ===>                                                 SCROLL ===> CSR
TRACE=> 000060 000059 000058 000056 000055 000050 000049 000048 000047 000046 0
     ------------------------------------------------------------------------
     00000000 HX    000000 02 R1                        000000000000000000.
     ------------------------------------------------------------------------
 0001 000096 900F 1000           00000    59          STM   R0,R15,0(R1)
 ---> 00009A EB0F 1040 0026      00040    60          STMH  R0,R15,64(R1)
 ---> 0000A0 9B0F 1080           00080    61          STAM  R0,R15,128(R1)
```

L or K Command (Display Data Item in a Keep Window)

For COBOL or PL/I programs, the L line command causes a data item to be displayed on the Intercept panel for any data items contained on the line where the L is typed. For assembler programs, the L line command displays the operands for the current instruction in register notation. This is commonly known as a *keep* window. For more information on LDI commands, see the LDI (or KEEP) command in the Control Commands section that follows. The number of L's you can specify is limited only by the number of data items displayed because of the screen size.

Cursor-sensitive keep: If you enter L or K and the cursor is placed on a COBOL or PL/I data item in that line, pressing Enter lists in the keep window only that data item and data items to the right of that item on the same line.

The LDI (or keep window) displays may be reset using the *remove all* command or the R line command.

Note: Elementary items are modifiable from within the keep window.

In the following panel the program has stopped at a breakpoint. To display all the data items in the MOVE statement, you must type a K on each of the lines on which the statement occurs. In this case, you must type K on lines 13400, 13410, 13500, and 13600. Pressing Enter causes the command to process.

```
CAMRCOB2 ---------- CA InterTest Batch NEXT BEFORE Intercept -----------------
COMMAND ===>                                                 SCROLL===> PAGE
TRACE=> 013400 013300 013200 013100 020100 020000 012900 012800 012700 012600
        7        CAMRCOB2        15.49.31        JUN 30,2014
 0001        012300 PROCEDURE DIVISION USING PARMS TEST-THREE TEST-FOUR PASS-
 0001        012400 DEMO-BEGIN.
 0001        012500     GO TO DEMO-INIT.
 0001        012600 DEMO-INIT.
 0001        012700     OPEN OUTPUT OUTPUT01.
 0001        012800     MOVE HD1 TO LINE-OUT.
 0001        012900     PERFORM WRITE-LINE.

U0001        013100 FORCING-THE-ABEND.
 0001        013200     MOVE MASK1  TO BIN1.
 0001        013300     MOVE ZERO   TO MASK1.
k--->        013400     MOVE MASK1  TO BIN1
k            013410                    BIN2
k            013500                    BIN3
k            013600                    BIN4.
 --->        013700 TABLE-MOVE.
U--->        013736       MOVE TABLE-FILL TO WA-TABLE-1.
 --->        013737 NUM-EDITED-TEST.
```

The following panel shows the keep window displays for data items MASK1, BIN1, BIN2, BIN3, and BIN4:

```
CAMRCOB2 ---------- CA InterTest Batch NEXT BEFORE Intercept -----------------
COMMAND ===>                                                 SCROLL===> PAGE
TRACE=> 013400 013300 013200 013100 020100 020000 012900 012800 012700 012600
      ------------------------------------------------------------------------
      00204DA0 AN    004000 03 MASK1                       00
      00204DB2 NB-S  004600 03 BIN1                        +00000001.
      00204DB6 NB-S  004700 03 BIN2                        +00000016.
      00204DBA NB-S  004800 03 BIN3                        +00000032.
      00204DBE NB-S  004900 03 BIN4                        +00000077.
      ------------------------------------------------------------------------
        7        CAMRCOB2        15.49.31        JUN 30,2014
 0001        012300 PROCEDURE DIVISION USING PARMS TEST-THREE TEST-FOUR PASS-
 0001        012400 DEMO-BEGIN.
 0001        012500     GO TO DEMO-INIT.
 0001        012600 DEMO-INIT.
 0001        012700     OPEN OUTPUT OUTPUT01.
 0001        012800     MOVE HD1 TO LINE-OUT.
 0001        012900     PERFORM WRITE-LINE.
U0001        013100 FORCING-THE-ABEND.
 0001        013200     MOVE MASK1  TO BIN1.
 0001        013300     MOVE ZERO   TO MASK1.
U--->        013400     MOVE MASK1  TO BIN1
             013410                    BIN2
```

O or X Command (Turn Off Breakpoints)

The O or X line command removes a breakpoint that is set for the line on which it is typed. A blank in the first field on a line also resets the breakpoint. The O line command should only be used on lines that contain a U or S in the first column (where a breakpoint is set). An O on any other line has no effect. The number of O commands that may be processed is limited only by the display screen size.

C, W, and V line commands are not displayed and therefore cannot be removed with an O or X.

Note: X is a synonym of the R line command when it is used in the keep window. When it is used on a source line, X is a synonym of the O line command.

In the following panel, you have entered an O line command on lines 13400 and 13736 by typing over the U's in the first column:

```
CAMRCOB2 ---------- CA InterTest Batch NEXT BEFORE Intercept -----------------
COMMAND ===>                                                 SCROLL===> PAGE
TRACE=> 013400 013300 013200 013100 020100 020000 012900 012800 012700 012600
         7        CAMRCOB2        15.49.31        JUN 30,2014
 0001        012300 PROCEDURE DIVISION USING PARMS TEST-THREE TEST-FOUR PASS-
 0001        012400 DEMO-BEGIN.
 0001        012500     GO TO DEMO-INIT.
 0001        012600 DEMO-INIT.
 0001        012700     OPEN OUTPUT OUTPUT01.
 0001        012800     MOVE HD1 TO LINE-OUT.
 0001        012900     PERFORM WRITE-LINE.

U0001        013100 FORCING-THE-ABEND.
 0001        013200     MOVE MASK1  TO BIN1.
 0001        013300     MOVE ZERO   TO MASK1.
o--->        013400     MOVE MASK1  TO BIN1
             013410                    BIN2
             013500                    BIN3
             013600                    BIN4.
 --->        013700 TABLE-MOVE.
o--->        013736       MOVE TABLE-FILL TO WA-TABLE-1.
 --->        013737 NUM-EDITED-TEST.
```

After you press Enter, the U's are no longer on the lines. This indicates that the breakpoints have been reset as shown in the following panel:

```
CAMRCOB2 ---------- CA InterTest Batch NEXT BEFORE Intercept -----------------
COMMAND ===>                                                 SCROLL===> PAGE
TRACE=> 013400 013300 013200 013100 020100 020000 012900 012800 012700 012600
         7        CAMRCOB2        15.49.31        JUN 30,2014
 0001        012300 PROCEDURE DIVISION USING PARMS TEST-THREE TEST-FOUR PASS-
 0001        012400 DEMO-BEGIN.
 0001        012500     GO TO DEMO-INIT.
 0001        012600 DEMO-INIT.
 0001        012700     OPEN OUTPUT OUTPUT01.
 0001        012800     MOVE HD1 TO LINE-OUT.
 0001        012900     PERFORM WRITE-LINE.

U0001        013100 FORCING-THE-ABEND.
 0001        013200     MOVE MASK1  TO BIN1.
 0001        013300     MOVE ZERO   TO MASK1.
 --->        013400     MOVE MASK1  TO BIN1
             013410                    BIN2
             013500                    BIN3
             013600                    BIN4.
 --->        013700 TABLE-MOVE.
 --->        013736       MOVE TABLE-FILL TO WA-TABLE-1.
 --->        013737 NUM-EDITED-TEST.
```

You can confirm that you reset the breakpoints by using the LISTAT command to produce the Breakpoint Status panel.

P Command (Print Display)

For COBOL or PL/I programs, the P line command writes the display in the session log for all the data items on the line where you typed the command. For assembler programs, the P line command prints the operands for the current instruction in register notation. The data is written as it would be displayed using the D line command. For a description of the Display Panel, see Data Display Panels in the chapter "Debugging Panels."

You can type the P line command on any line in the program listing. You can enter multiple P commands on different lines.

For further description and a sample of the session log, see Session Log Facility (Review Debugging Session) at the end of this chapter.

R Command (Reset Data Item Display)

The R line command resets a data item display from an LDI command, an L line command, an LDX command, or an H line command. Place the R line command on a line on the program listing to reset all the data items on that line for which List Data Item displays are present, or place it directly on a listed data item display to reset that display.

Note: X is a synonym of the R line command when it is used in the keep window. When it is used on a source line, X is a synonym of the O line command.

In the following panel, an LDI has been done for MASK1 and BIN1, and an LDX has been done for BIN2. By placing an R on the data item display for BIN1 and line 13410 containing BIN2, these data items are reset after you press Enter.

```
CAMRCOB2 ---------- CA InterTest Batch NEXT BEFORE Intercept -----------------
COMMAND ===>                                                  SCROLL===> PAGE
TRACE=> 013400 013300 013200 013100 020100 020000 012900 012800 012700 012600
      ------------------------------------------------------------------------
      00204DA0 AN    004000 03 MASK1                        00
r     00204DB2 NB-S  004600 03 BIN1                         +00000001.
      00204DB6 NB-S  004700 03 BIN2                         +00000016.
               DATA OCCUPIES 0004 BYTES OF STORAGE     ZONES 0001
                                                     NUMERICS 0000
      ------------------------------------------------------------------------
         7        CAMRCOB2        15.49.31        JUN 30,2014
 0001        012300 PROCEDURE DIVISION USING PARMS TEST-THREE TEST-FOUR PASS-
 0001        012400 DEMO-BEGIN.
 0001        012500     GO TO DEMO-INIT.
 0001        012600 DEMO-INIT.
 0001        012700     OPEN OUTPUT OUTPUT01.
 0001        012800     MOVE HD1 TO LINE-OUT.
 0001        012900     PERFORM WRITE-LINE.

U0001        013100 FORCING-THE-ABEND.
 0001        013200     MOVE MASK1  TO BIN1.
 0001        013300     MOVE ZERO   TO MASK1.

 --->        013400     MOVE MASK1  TO BIN1
r            013410                    BIN2
```

The following panel shows the results of the R line commands. The BIN1 and BIN2 displays have been reset, leaving only the display for MASK1:

```
CAMRCOB2 ---------- CA InterTest Batch NEXT BEFORE Intercept -----------------
COMMAND ===>                                                 SCROLL===> PAGE
TRACE=> 013400 013300 013200 013100 020100 020000 012900 012800 012700 012600
     ------------------------------------------------------------------------
     00204DA0 AN    004000 03 MASK1                        00
     ------------------------------------------------------------------------
        7        CAMRCOB2        15.49.31        JUN 30,2014
 0001        012300 PROCEDURE DIVISION USING PARMS TEST-THREE TEST-FOUR PASS-
 0001        012400 DEMO-BEGIN.
 0001        012500     GO TO DEMO-INIT.
 0001        012600 DEMO-INIT.
 0001        012700     OPEN OUTPUT OUTPUT01.
 0001        012800     MOVE HD1 TO LINE-OUT.
 0001        012900     PERFORM WRITE-LINE.

U0001        013100 FORCING-THE-ABEND.
 0001        013200     MOVE MASK1  TO BIN1.
 0001        013300     MOVE ZERO   TO MASK1.

 --->        013400     MOVE MASK1  TO BIN1
             013410                    BIN2
             013500                    BIN3
             013600                    BIN4.
 --->        013700 TABLE-MOVE.
 --->        013736       MOVE TABLE-FILL TO WA-TABLE-1.
```

S Command (Skip this Statement)

The S (Skip) line command when placed directly next to a statement causes the statement to be skipped when the program is executing. The S remains on the line to show you which statements are not executed. Skip is considered an unconditional breakpoint.

You may type other commands over the S. To cancel a Skip line command use a blank, X, or O line command. If you use OFF ALL to remove all breakpoints, all skip conditions are also removed.

Important! When a program has been optimized, or when the statement being skipped is any of the following, the S (Skip) line command may have undesired results:

- The last statement in the program
- The last statement in a performed paragraph
- Immediately followed by an ELSE statement

For more details, see the SKIP Command (Skip a Statement) in the chapter “Debugging Commands”.

The following panel shows a skip breakpoint being set at statement 21010. The S causes the call to the program A1 to be skipped as the program is executed:

```
CAMRCOB2 ---------- CA InterTest Batch *INITIAL* Intercept -------------------
COMMAND ===>                                                 SCROLL===> HALF
TRACE=> 020400
        8                          17.28.55       MAR 10,2014
 *-->        020400 PROCEDURE DIVISION USING PARMS TEST-THREE TEST-FOUR PASS-
 *-->        020500 DEMO-BEGIN.
 *-->        020600     GO TO DEMO-INIT.
 *-->        020700 DEMO-INIT.
 *-->        020800     OPEN OUTPUT OUTPUT01.
 *-->        020900     MOVE HD1 TO LINE-OUT.
 *-->        021000     PERFORM WRITE-LINE.
S*-->        021010     CALL 'A1'.
 *-->        021020     CALL 'A2'.

U*-->        021200 FORCING-THE-ABEND.
 *-->        021300     MOVE MASK1  TO BIN1.
 *-->        021310     DISPLAY MASK1 BIN1 BIN2 BIN3 BIN4.
 *-->        021400     MOVE ZERO   TO MASK1.
 *-->        021500     MOVE MASK1  TO BIN1
             021600                   BIN2
             021700                   BIN3
             021800                   BIN4.
 *-->        021900     IF BIN1 = BIN2
             022000       NEXT SENTENCE.
```

The following panel shows what the program looks like after you have used the GO command to resume program execution. The program is intercepted by the breakpoint on line 21200. Note that there is no frequency counter on statement 21010. This statement was not executed.

```
CAMRCOB2 ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
COMMAND ===>                                                 SCROLL===> HALF
TRACE=> 021200 021020 021000 020900 020800 020700 020600 020500 020400
        8                          17.28.55       MAR 10,2014
 0001        020400 PROCEDURE DIVISION USING PARMS TEST-THREE TEST-FOUR PASS-
 0001        020500 DEMO-BEGIN.
 0001        020600     GO TO DEMO-INIT.
 0001        020700 DEMO-INIT.
 0001        020800     OPEN OUTPUT OUTPUT01.
 0001        020900     MOVE HD1 TO LINE-OUT.
 0001        021000     PERFORM WRITE-LINE.
S--->        021010     CALL 'A1'.
 0001        021020     CALL 'A2'.

U--->        021200    FORCING-THE-ABEND.
 --->        021300     MOVE MASK1  TO BIN1.
 --->        021310     DISPLAY MASK1 BIN1 BIN2 BIN3 BIN4.
 --->        021400     MOVE ZERO   TO MASK1.
 --->        021500     MOVE MASK1  TO BIN1
             021600                   BIN2
             021700                   BIN3
             021800                   BIN4.
 --->        021900     IF BIN1 = BIN2
             022000       NEXT SENTENCE.
```

V Command (Set Variable Change Breakpoint)

The V line command sets a variable change breakpoint at the specified line. A when-name is automatically generated. The name is VR*nnnnnn* where n=line number of the breakpoint. This means you can have only one conditional set by the V line command per line. The breakpoint is triggered before the line is executed when the application determines that the value of a variable in the line changes. The condition is *not* checked before each verb. This command is only valid for COBOL or PL/I programs.

Cursor-sensitive variable change breakpoint: If you enter V and the cursor is placed on a COBOL or PL/I data item, pressing Enter designates that item as the variable change item.

The following panel shows variable change breakpoints being set on lines 1149 and 1154 of the program. You must press Enter after entering the breakpoints in order to set them:

```
CAMRCOB2 ---------- CA InterTest Batch NEXT BEFORE Intercept -----------------
Command ===>                                                 Scroll===> CUR
TRACE=> 001151 001150 001149 001148 001162 001155 001154 001153 001151 001150 0
 U0001   001141               ORDER-CALC.
  0001   001142                  MOVE CUSTOMER-VALUES TO CUSTOMER-ORDER.
  0001   001143                  MOVE ZEROES TO ORDER-SUB-TOTAL, SUB-6, LOOP-O
  0001   001144                  PERFORM TOTAL-ORDER THRU TOTAL-ORDER-EX
         001145                       UNTIL SUB-6 EQUAL 5.
 U--->   001146               ORDER-CALC-EX.
  --->   001147                  GO TO OPTIONS.
  0101   001148               TOTAL-ORDER.
 v0101   001149                  ADD 1 TO SUB-6.
  0101   001150                  ADD 1 TO LOOP-OUT.
  0100   001151                  MULTIPLY UNIT-PRICE(SUB-6) BY NO-OF-WIDGETS(S
         001152                       GIVING ITEM-TOTAL.
  0100   001153                  ADD ITEM-TOTAL TO ORDER-SUB-TOTAL.
 v0100   001154                  MOVE 1 TO SUB-6.
       PP 5668-958 IBM VS COBOL II Release 3.2 09/05/90              CAMRCOB2
       LineID  PL SL  ----+-*A-1-B--+----2----+----3----+----4----+----5----+-
```

+ (plus) Command (Increment a Subscript)

Use the plus (+) line command to increment the value of a subscript on a displayed data item. Display the data item using the LDI or LDX control commands or the L or H line commands. The format of the command is +*nnn* where *nnn* is any number from 1 to 999. The subscript is incremented by the amount specified by *nnn*. The default is 1, and this is the value that is used if *nnn* is not specified. More than one subscript may be incremented or decremented at the same time. This command is only valid for COBOL or PL/I programs.

The following panel shows a data display for WA-T1-3A(BIN1,4,1) of WA-TABLE-1 of WA-LEVEL1. Currently, WA-T1-3A(2,4,1) is displayed. Note the address and the value. The first subscript (BIN1) and the third subscript are incremented by the value of 2.

```
CAMRCOB2 ---------- CA InterTest Batch UNCOND BEFORE Intercept --------------
COMMAND ===>                                                 SCROLL===> HALF
TRACE=> 022300 022200 022100 021900 021500 021400 021310 021300 021200 021020
        ------------------------------------------------------------------------
        00262E08 ND    016800 06 WA-T1-3A                   01.
 +2                             BIN1  (2)
                                (4)
 +2                             (1)
                              OF WA-TABLE-1
                              OF WA-LEVEL1
        ------------------------------------------------------------------------

 0001         021200 FORCING-THE-ABEND.
 0001         021300     MOVE MASK1  TO BIN1.
 0001         021310     DISPLAY MASK1 BIN1 BIN2 BIN3 BIN4.
 0001         021400     MOVE ZERO   TO MASK1.
 0001         021500     MOVE MASK1  TO BIN1
              021600                    BIN2
              021700                    BIN3
              021800                    BIN4.
 0001         021900     IF BIN1 = BIN2
              022000       NEXT SENTENCE.
 0001         022100 TABLE-MOVE.
 0001         022200       MOVE TABLE-FILL TO WA-TABLE-1 OF WA-LEVEL1.
U--->         022300 NUM-EDITED-TEST.
 --->         022310       MOVE 'XP' TO ED-ALPHA.
```

Pressing Enter causes the line commands to be executed. The value of BIN1 and the value of the third subscript have been incremented by 2, so that WA-T1-3A(4,4,3) is now the table item displayed. Note that by incrementing the subscript variable BIN1, the value of BIN1 has been changed to 4 as shown in the following panel:

```
CAMRCOB2 ---------- CA InterTest Batch UNCOND BEFORE Intercept --------------
COMMAND ===>                                                 SCROLL===> HALF
TRACE=> 022300 022200 022100 021900 021500 021400 021310 021300 021200 021020
      ------------------------------------------------------------------------
      00262E6C ND    016800 06 WA-T1-3A                   03.
                               BIN1  (4)
                              (4)
                              (3)
                            OF WA-TABLE-1
                            OF WA-LEVEL1
      ------------------------------------------------------------------------

 0001         021200 FORCING-THE-ABEND.
 0001         021300     MOVE MASK1  TO BIN1.
 0001         021310     DISPLAY MASK1 BIN1 BIN2 BIN3 BIN4.
 0001         021400     MOVE ZERO   TO MASK1.
 0001         021500     MOVE MASK1  TO BIN1
              021600                    BIN2
              021700                    BIN3
              021800                    BIN4.
 0001         021900     IF BIN1 = BIN2
              022000       NEXT SENTENCE.
 0001         022100 TABLE-MOVE.
 0001         022200       MOVE TABLE-FILL TO WA-TABLE-1 OF WA-LEVEL1.
U--->         022300 NUM-EDITED-TEST.
 --->         022310       MOVE 'XP' TO ED-ALPHA.
```

- (minus) Command (Decrement a Subscript)

Use the minus (-) command to decrement the value of a subscript on a displayed data item. Display the data item using the LDI or LDX control commands or the L or H line commands. The format of the command is *-nnn* where *nnn* is any number from 1 to 999. The subscript is decremented by the amount specified by *nnn*. The default is 1, and this is the value that is used if *nnn* is not specified. You may increment or decrement more than one subscript at the same time. This command is only valid for COBOL or PL/I programs.

The following panel shows a data display for WA-T1-3A(BIN1,4,3) of WA-TABLE-1 of WA-LEVEL1. Currently, WA-T1-3A(4,4,3) is displayed. Note the address and the value. The first subscript (BIN1) is decremented by 3, and the second and third subscripts are decremented by 1:

```
CAMRCOB2 ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
COMMAND ===>                                                  SCROLL===> HALF
TRACE=> 022300 022200 022100 021900 021500 021400 021310 021300 021200 021020
     ------------------------------------------------------------------------
     00262E6C ND    016800 06 WA-T1-3A                  03.
 -3                              BIN1  (4)
 -1                              (4)
 -1                              (3)
                            OF WA-TABLE-1
                            OF WA-LEVEL1
     ------------------------------------------------------------------------

 0001         021200 FORCING-THE-ABEND.
 0001         021300     MOVE MASK1  TO BIN1.
 0001         021310     DISPLAY MASK1 BIN1 BIN2 BIN3 BIN4.
 0001         021400     MOVE ZERO   TO MASK1.
 0001         021500     MOVE MASK1  TO BIN1
              021600                    BIN2
              021700                    BIN3
              021800                    BIN4.
 0001         021900     IF BIN1 = BIN2
              022000       NEXT SENTENCE.
 0001         022100 TABLE-MOVE.
 0001         022200       MOVE TABLE-FILL TO WA-TABLE-1 OF WA-LEVEL1.
U--->         022300 NUM-EDITED-TEST.
 --->         022310       MOVE 'XP' TO ED-ALPHA.
```

Pressing Enter causes the line commands to be executed. The value of BIN1 has been decremented by 3, and the second and third subscripts have been decremented by 1, so that WA-T1-3A(1,3,2) is now the table item displayed.

Note that by decrementing the subscript variable BIN1, the value of BIN1 has been changed to 1 as shown in the following panel:

```
CAMRCOB2 ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
COMMAND ===>                                                 SCROLL===> HALF
TRACE=> 022300 022200 022100 021900 021500 021400 021310 021300 021200 021020
        ------------------------------------------------------------------------
        00262DCF ND    016800 06 WA-T1-3A                    02.
                                 BIN1  (1)
                                 (3)
                                 (2)
                               OF WA-TABLE-1
                               OF WA-LEVEL1
        ------------------------------------------------------------------------

 0001         021200 FORCING-THE-ABEND.
 0001         021300     MOVE MASK1  TO BIN1.
 0001         021310     DISPLAY MASK1 BIN1 BIN2 BIN3 BIN4.
 0001         021400     MOVE ZERO   TO MASK1.
 0001         021500     MOVE MASK1  TO BIN1
              021600                    BIN2
              021700                    BIN3
              021800                    BIN4.
 0001         021900     IF BIN1 = BIN2
              022000       NEXT SENTENCE.
 0001         022100 TABLE-MOVE.
 0001         022200       MOVE TABLE-FILL TO WA-TABLE-1 OF WA-LEVEL1.
U0001         022300 NUM-EDITED-TEST.
```

<[nnnn] (Scroll Left) Command

The scroll left (<) command is used to scroll data in the keep window left. The format of the command is <[*nnnn*] where *nnnn* is any number from 1 to 9999. If the data length is greater than the display area, the data is scrolled the number of bytes indicated by *nnnn*. Data cannot be scrolled beyond the length of the data item. If the *nnnn* value is greater than the length of the data item, the display is positioned at the last *n* number of bytes that fit in the display area.

The following panel shows a data item scrolled left 5 bytes:

```
CAMRCOB2 ---------- CA InterTest Batch *INITIAL* Intercept ------------------
COMMAND ===>                                                 SCROLL ===> CSR
TRACE=>
      ------------------------------------------------------------------------
      1F0A9630       000838 01 TRL-TABLE
      1F0A9635 AN    000839 03 FILLER              (+0005) EFGHIJKLMNOPQRSTUV
      1F0A964B AN    000841 03 FILLER                      1234567890
      1F0A9655 AN    000843 03 FILLER                      -+=.,;:#*/()@&%$¢?
      ------------------------------------------------------------------------
      000875        PROCEDURE DIVISION USING PARAMETER-AREA.
 *--> 000876        START-PROGRAM.
 *--> 000877            OPEN OUTPUT REPORT-OUT.
      000878       * -------------------------------------------*
      000879       * Welcome to the CA InterTest Batch COBOL II     *
      000880       * Demonstration Program. You are now at the      *
      000881       * Initial Breakpoint panel which is displayed    *
      000882       * upon entry to the first program to be tested.  *
      000883       * At this point, other breakpoints can be set    *
      000884       * and control commands can be issued before  the *
      000885       * program is executed.                           *
      000886       * -------------------------------------------*
 *--> 000887            MOVE  R1498-TIMEI TO R1498-TIME.
 *--> 000888            PERFORM TIME-EDIT THRU TIME-EDIT-EX.
      000889       *
 *--> 000890            MOVE  R1498-TIMEO TO R1498-TIME.
 *--> 000891            PERFORM TIME-EDIT THRU TIME-EDIT-EX.
 *--> 000892            GO TO DISPLAY-1ST-PANEL.
Note: Scrolling starts from relative zero. In the previous example the displayed
character “E” is at real position 6 (+0005 relative to 0).
```

>[nnnn] (Scroll Right) Command

The scroll right (>) command is used to scroll data in the keep window right. The format of the command is >[*nnnn*] where *nnnn* is any number from 1 to 9999. If the data length is greater than the display area, the data is scrolled the number of bytes indicated by *nnnn*. Data cannot be scrolled beyond the length of the data item. If the *nnnn* value is greater than the length of the data item, the display is positioned at the first *n* number of bytes that fit in the display area.

The following panel shows a data item scrolled right 3 bytes:

```
CAMRCOB2 ---------- CA InterTest Batch *INITIAL* Intercept ------------------
COMMAND ===>                                                SCROLL ===> CSR
TRACE=>
      ------------------------------------------------------------------------
      1F0A9630       000838 01 TRL-TABLE
      1F0A9632 AN    000839 03 FILLER              (+0002) BCDEFGHIJKLMNOPQRS
      1F0A964B AN    000841 03 FILLER                      1234567890
      1F0A9655 AN    000843 03 FILLER                      -+=.,;:#*/()@&%$¢?
      ------------------------------------------------------------------------
      000875        PROCEDURE DIVISION USING PARAMETER-AREA.
 *--> 000876        START-PROGRAM.
 *--> 000877            OPEN OUTPUT REPORT-OUT.
      000878       * ---------------------------------------------*
      000879       * Welcome to the CA InterTest Batch COBOL II    *
      000880       * Demonstration Program. You are now at the     *
      000881       * Initial Breakpoint panel which is displayed   *
      000882       * upon entry to the first program to be tested. *
      000883       * At this point, other breakpoints can be set   *
      000884       * and control commands can be issued before  the *
      000885       * program is executed.                          *
      000886       * ---------------------------------------------*
 *--> 000887            MOVE  R1498-TIMEI TO R1498-TIME.
 *--> 000888            PERFORM TIME-EDIT THRU TIME-EDIT-EX.
      000889       *
 *--> 000890            MOVE  R1498-TIMEO TO R1498-TIME.
 *--> 000891            PERFORM TIME-EDIT THRU TIME-EDIT-EX.
 *--> 000892            GO TO DISPLAY-1ST-PANEL.
```

Note: Scrolling starts from relative zero. In the previous example the displayed character “B” is at real position 3 (+0002 relative to 0)

Control Commands

Control commands control the test session. Enter them on the command line of the Intercept panel. Enter multiple control commands on the command line by separating each command with a semicolon (;).

Assembler Indirect Addressing Note: Some control commands can accept an expression in *register* notation when debugging an assembler program. The format of an expression is as follows:

```
<base <suffix>> <<offset<suffix>>...>
```

Operands	Description
base	hexadecimal address, or register number (that is, R1 for general register 1), or a data name
suffix	% for indirect addressing in 24 bit mode ? for indirect addressing in 31 bit mode ! for indirect addressing in 64 bit mode

Operands	Description
offset	plus (+) or minus (-) a hexadecimal value up to 4 characters long

Note: When starting an expression with a hexadecimal address, it must start with a plus (+) sign.

The following table shows a list of control commands available when using your product:

Command	Definition
ADVANCE	Follows the trace pointers forward.
AT	Sets an unconditional breakpoint (same as UNCOND).
CORE	Displays memory for the debugging region.
COUNTS	Enables frequency counting, reset counters, display counters.
CS	Highlights the current statement.
DATAMON	Enables the data monitoring feature.
DDALLOC	Allocates a DD.
DDFREE	Frees a DD.
DDQ	Queries a DD.
DISPLAY	Formats a data item or the COBOL Working Storage section.
DROP	Drops the symbolic name.
DROPUSE	Removes a using that was set with the USING command.
DUMP	Terminates testing with a dump.
EQUATE	Specifies symbolic name.
EXSUM	Provides the execution summary display.
FILES	Displays COBOL FD status.
FM	Starts CA File Master Plus.
FREQ	Controls verb execution counts.
GO	Resumes execution of the application.
INCLUDE	Executes a set of application commands.
LDA (or AUTOKEEP)	Displays all data items referenced by the current statement.
LDI (or KEEP)	Lists data item.

Command	Definition
LDX (or KEEPX)	Lists data item in hexadecimal.
LEQ	Lists equated symbolic names.
LINKAGE	Formats the COBOL LINKAGE section.
LISTAT	Lists breakpoints.
LISTBP	Displays the BREAKPOINT panel.
LISTLBL	Lists labels added with the .label command.
LISTUSE	Displays the USING STATUS panel.
LISTWHEN	Lists WHEN conditions.
MAP	Invokes the Region Map Display panel.
NEXT	Executes the next *n* verbs.
NOFREQ	Removes frequency counters.
OFF (or OFFU)	Removes unconditional breakpoints.
OFFWN (OFFC)	Deletes conditional breakpoints.
POINT	Directs PREVIOUS and ADVANCE pointer.
PREV	Follows trace pointers in reverse.
PS	Prints stream to session log.
QUALIFY	Sets current program id.
QUIT	Terminates the test session.
RDI	Resets data item (or REMOVE).
RECORDS	Formats data in the COBOL FILE section.
REFRESH	Displays updated split screen values.
REGS	Displays registers for assembler programs.
RESET	Removes column numbers from display.
RESTART	Restarts a batch link debugging step.
RUN	Removes breakpoints and execute.
SDWA	Displays the system diagnostic work area.
SET	Changes value of a data item.
SKIP	Skips a statement.
SLOW	Observes execution a verb at a time.

Command	Definition
SNAP	Produces a CA Optimizer, CA Optimizer/II, or CA SymDump Batch SNAP report.
SPEED	Accelerates program testing.
STEP	Sets count for GO command.
SUSPEND	Detaches a terminal from a batch link debugging session without terminating the session.
TRACE	Controls/displays program trace entries.
TRACE SOURCE	Controls/displays program trace entries and their corresponding source code.
TRP	Displays traced paragraph entries.
UNCOND	Sets an unconditional breakpoint (same as AT).
USING	Sets an assembler base register for a DSECT that currently has no active USING.
WHEN (or COND)	Sets conditional breakpoints (or COND).
*	Adds a comment to the session log.
/clist	Executes CLIST CAMRCMD, called with parameter *clist*.

ADVANCE Command (Follow Trace Pointers Forward)

Referencing the trace table, the ADVANCE command highlights and displays the next statement that was executed. This feature lets the programmer visually follow in a forward sequence the statements that have been executed. This command is used along with the PREVIOUS command.

Syntax

Use the following syntax for ADVANCE:

```
ADVANCE <nnn|P nn>
```

You may enter this command in its abbreviated form: A.

Variables

Use the following variables with the ADVANCE command:

nnn Specifies the number of executed statements for the session to move forward. The default is one.

P Specifies that the next executed paragraph, procedure, or label is displayed.

nn Specifies the number of paragraphs, procedures, or labels for the session to move forward. The default is one.

Usage Notes

The trace facility must be turned on for the ADVANCE command to function. The trace facility is turned on by default when you first invoke the application. For further information on turning the trace facility on and off, see the TRACE command later in this chapter.

The ADVANCE command visually displays the forward execution of the program using the trace table entries. It does not actually execute the program.

The ADVANCE command does not affect the point at which a program resumes execution. Execution is resumed by issuing a GO command.

If you have not used the PREV command, the ADVANCE P command scrolls the program to the current statement, highlights it, and displays END OF TRACE TABLE in the upper right-hand corner. Use this to return the program to the current statement after scrolling through the listing. The control command CS also displays and highlights the current statement.

If data monitoring was in effect when the statement highlighted from the ADVANCE command was executed, the values displayed for your data items are from the time before that statement was executed.

AT (or UNCOND) Command (Set an Unconditional Breakpoint)

Use the AT command to set breakpoints from the Intercept panel or to display the Breakpoint panel. The UNCOND command is a synonym of AT.

If you enter the AT or UNCOND command with operands in the Command field of an Intercept panel, the breakpoint is set.

If you enter AT or UNCOND without operands, the Breakpoint panel is displayed.

Syntax

Use the following syntax for AT, UNCOND or U:

```
AT <<statement-number|paragraph-name|procedure-name|label-name> <AFTER|BEFORE>
<(subcommand-list)>>
<<ALL|program-name> <ENTRY|EXIT|LABEL|DBCALL>  <(subcommand-list)>>
```

Variables

See the following variables with the AT or UNCOND commands:

statement-number	Specifies a statement-number in the program where a breakpoint is set.
paragraph-name	Specifies a paragraph-name in the COBOL program where a breakpoint is set.
procedure-name	Specifies a procedure-name in the PL/I program where a breakpoint is set.
label-name	Specifies label-name in the assembler or PL/I program where a breakpoint is set.
ENTRY	For COBOL programs, specifies that a breakpoint occurs each time the currently qualified program begins execution. For PL/I and assembler programs, specifies that a breakpoint occurs the first time the currently qualified program is entered.
EXIT	Specifies that a breakpoint occurs each time the currently qualified COBOL program finishes execution.
LABEL	Specifies that a breakpoint occurs each time the currently qualified program executes a paragraph, procedure, or label.
DBCALL	Specifies that a breakpoint occurs each time the currently qualified program makes a call to a database, such as DB2 or IMS
ALL ENTRY	Specifies that a breakpoint occurs each time a monitored COBOL program in the run unit begins execution and the first time a monitored assembler or PL/I program in the run unit begins execution.
ALL EXIT	Specifies that a breakpoint occurs each time any monitored COBOL program in the run unit finishes execution.
ALL LABEL	Specifies that a breakpoint occurs each time a monitored program executes a label.
ALL DBCALL	Specifies that a breakpoint occurs each time a monitored program makes a call to a database, such as DB2 or IMS.
AFTER	Specifies that the breakpoint should be triggered after the statement is executed.
BEFORE	Specifies that the breakpoint should be triggered before the statement is executed. This is the default.

subcommand-list A list of application commands, separated by semicolons, to be executed when the breakpoint is encountered during the test. When testing under the ISPF dialog manager, you may enter only one subcommand unless the ISPF command delimiter character is changed to something other than the semicolon. This can be done under option 0.1 of the ISPF main menu.

Usage Notes

Set a breakpoint by the paragraph name or by the line number of the paragraph. Sample statements to set breakpoints by paragraph name are shown next:

```
041400 0000-INITIALIZATION SECTION.
041500
041600 MOVE +20 TO VARIABLE-LENGTH-1.
041700 MOVE +25 TO VARIABLE-LENGTH-2.
041800 MOVE +30 TO VARIABLE-LENGTH-3.
```

Set a breakpoint on the first statement of the example by either of the following AT commands:

```
AT 41400 or AT 0000-INITIALIZATION
```

Set a breakpoint with a list of application commands to be executed at the breakpoint. The commands are executed one at a time until the program resumes execution or until a transfer of panels occurs, at which point the remaining commands are ignored. Commands should be in parentheses, separated by semicolons, as follows:

```
AT 41700 (SET VARIABLE-LENGTH-2 = 50;GO 41800)
```

In this example, the program stops before executing statement 41700, and VARIABLE-LENGTH-2 is set to 50. The GO 41800 causes the program to resume execution at statement 41800 so the contents of VARIABLE-LENGTH-2 are not changed to +25. You can use commands like these to temporarily modify the execution flow of the program.

ISPF Note

When running under ISPF, specify only one command in the subcommand list because of restricted use of the semicolon, unless the ISPF command delimiter character has been changed. You can do this in option 0.1 of the ISPF main menu.

CORE Command (Display Memory)

The CORE command invokes the Display-Alter Memory panel. This panel lets you modify data based on its address. For more information, see the chapter "Core and the Map Options."

Syntax

Use the following syntax for CORE:

```
CORE <data-item|paragraph-name|address|expression>
```

Usage Notes

If you are navigating the trace table and data monitoring was in effect for the currently highlighted statement, you will see a warning panel. The CORE command shows only the current values of data items and not previous values. Previous values of data items cannot be modified.

COUNTS Command (Control Frequency Counting)

Use the COUNTS command to enable or disable frequency counting, reset frequency counters, and control the display of the frequency counters.

When enabled, frequency counting causes the debugger to count the number of times each statement is executed. When the display is active, the frequency counters appear on the Intercept panel in the left margin.

Syntax

Use the following syntax for COUNTS:

```
COUNTS <ON|OFF> <SHOW|NOSHOW> <RESET|RESET ALL|RESET progid>
```

Variables

Use the following variables with the COUNTS command:

ON	Enables frequency counting
OFF	Disables frequency counting
SHOW	Causes the frequency counters to be displayed on the Intercept panel
NOSHOW	Suppresses the display of the frequency counters on the Intercept panel

RESET	Clears to zero all of the counters for the current program
RESET ALL	Clears to zero all of the counters for all of the programs
RESET progid	Clears to zero all of the counters for the specified program

Usage Notes

When specifying COUNTS ON or OFF, you may also specify SHOW or NOSHOW as a second argument.

If frequency counting is enabled but the display is disabled, counting continues even though no counts are displayed.

CS Command (Current Statement)

The Current Statement (CS) command displays and highlights the current statement. This is especially useful after scrolling through the listing and any time the next statement to be executed is no longer displayed. You can also change the current statement by a passing statement number.

Syntax

Use the following syntax for CS:

```
CS <statement-number|paragraph-name|procedure-name|label-name>
```

Usage Notes

When using CS to modify the flow of execution of a COBOL program, be careful to avoid entering a paragraph and bypassing the PERFORM statement. When the exit to the paragraph is reached, a USER 519 abend may be issued because the references to the calling statement are not defined.

When using CS to modify the flow of execution of a PL/I program, be careful to avoid entering a block (that is, procedure, BEGIN block, or ON-unit) other than at the start of the block, or exiting a block other than at the END statement for the block, as it causes unpredictable results.

DATAMON (Enable Data Monitoring)

The DATAMON command enables data monitoring for the specified COBOL program. When this feature is used in conjunction with the trace navigation commands (PREV, ADV, POINT), you see past data values when displaying data.

Syntax

Use the following syntax for DATAMON:

```
DATAMON ON|OFF <pgmid>
```

You can use this command in its abbreviated form: DM.

Usage Notes

Enabling DATAMON lets you view past data values when navigating the trace table. If DATAMON was in effect for the statement highlighted by the trace navigation command (PREV, ADV, POINT), the data values displayed in the KEEP and DISPLAY commands will be past data values captured from a point in time *before* a COBOL statement referencing them is executed. These values are protected as you may not be able to update data while viewing past data values. For this reason, the SET command is disabled in this situation. When using the CORE command, a warning panel is shown indicating that the current data is shown for viewing and updating, not the past data values. Also, use of the + and - line commands are disabled when incrementing or decrementing a subscript symbolically.

The DATAMON command can be CPU intensive and it requires a significant amount of storage. The DATAMON command is only valid for COBOL programs.

DDALLOC Command (Allocate a DD)

The DDALLOC command displays the DDALLOC panel. From this panel, you allocate a DD to DUMMY, a SYSOUT class, or a data set. Note that this feature works in both batch link and foreground testing. Foreground users can also split their screen and use Option 3, ALLOCATIONS, to perform their allocations.

Syntax

Use the following syntax for DDALLOC:

```
DDALLOC <ddname>
```

You can enter this command in its abbreviated form: DDA.

DDFREE Command (Free a DD)

The DDFREE command lets you free a specified DD.

Syntax

Use the following syntax for DDFREE:

```
DDFREE ddname
```

You can enter this command in its abbreviated form: DDF.

DDQ Command (Query a DD)

The DDQ command invokes the DDName Query panel. The panel displays information about the allocated DD's.

Syntax

Use the following syntax for DDQ:

```
DDQ <ddname>
```

DISPLAY Command (Format Data or the COBOL Working Storage Section)

The DISPLAY command invokes the Display panel. You can modify the data values on this display.

Syntax

Use the following syntax for DISPLAY:

```
DISPLAY <data-item|expression <data-item> ...> <PRINT>
```

You can enter this command in its abbreviated form: D.

Note: Data for variables are case-sensitive.

Variables

Use the following variables with the DISPLAY command:

data item	Causes a display of the data item, any subordinate level data items, and their data values. You may enter more than one data item.
expression	For assembler programs, enter an expression in register notation.

PRINT Causes the data displayed by the command to be written to the session log.

Usage Notes

Consider the following COBOL WORKING-STORAGE definition:

```
WORKING-STORAGE SECTION.
01 WORKING-STORAGE-AREA.
          05 CONVERSION-AREA.
          10 CONV-DATE      PIC 9(7) VALUE ZERO.
          10 CONV-TIME      PIC 9(6)V999 VALUE ZERO.
          05 SELECTION CRITERIA.
          10 SELECT-START   PIC S9(15)V999 COMP.
          10 SELECT-END     PIC S9(15)V999 COMP.
```

If you enter the DISPLAY command without operands, it formats and displays all of the previous data items and their data values.

The command D CONVERSION-AREA would format and display CONVERSION-AREA, CONV-DATE, and CONV-TIME.

The command D SELECT-START would format and display only the data item SELECT-START.

Note: If data monitoring was in effect, data item values displayed are those captured at a point in time bef*ore th*e COBOL statement, positioned by Prev or Adv command is executed.

If a data item is modified on the other side of a split screen, you can update the Display panel to reflect any changes made by typing **REFRESH** on the command line.

Note: The data-item parameter may be a qualified data item, as in the following examples:

COBOL: Use the command D EMPLOYEE-NO IN MASTER-RECORD to refer to the EMPLOYEE-NO in MASTER-RECORD if EMPLOYEE-NO is not unique.

PL/I: Use the command D MASTER.RECORD.EMPLOYEE_NO to refer to the EMPLOYEE_NO in MASTER_RECORD if EMPLOYEE_NO is not unique.

Use the command D NEXT_ELEM_PTR->ELEMENT to refer to the allocation of based variable ELEMENT that is pointed to by NEXT_ELEM_PTR.

For assembler or PL/I programs, you need a data-item or expression operand.

If you are navigating the trace table and data monitoring was in effect for the currently highlighted statement, the values displayed for your data items are from the time before that statement was executed.

DROP Command (DROP Symbolic Name)

Use the DROP command to remove the symbolic name for a data item that has been set by the EQUATE command. View the symbolic names using the LEQ command or set by using the EQUATE command. DROP without operands causes the Equate panel to be displayed. For more information on the EQUATE command, see the Breakpoint Panels in the chapter "Debugging Panels."

Syntax

Use the following syntax for DROP:

```
DROP <symbolic-name>
```

You can enter this command in its abbreviated form: DR.

Variables

Use the following variable with the DROP command:

symbolic-name User-defined symbolic name for a data item.

DROPUSE Command (Remove a Using)

The DROPUSE command removes a using that was set with the USING command. This command is only valid when debugging an assembler program.

Syntax

Use the following syntax for DROPUSE:

```
DROPUSE dsect|csect
```

Variables

Use the following variables with the DROPUSE command:

dsect Assembler DSECT that you wish to map

csect Assembler CSECT that you wish to map

You can enter this command in its abbreviated form: DROPU.

DUMP Command (Terminate Testing with a Dump)

Use the DUMP command to terminate the testing session with a dump. This command only works when the application being tested has previously abended.

Syntax

Use the following syntax for DUMP:

```
DUMP
```

Usage Notes

To use the DUMP command, you must pre-allocate an MVS dump data set (SYSMDUMP). Allocate the dump data set from Option 3 (ISPF) - ALLOCATION, of the Primary Option Menu; from ISPF option 6 using the TSO allocate command; or by pre-allocating a dump data set in the CLIST. The following example is a sample allocation as it appears in a CLIST:

```
CONTROL NOMSG
FREE DD(SYSMDUMP)
DEL yourdsn
CONTROL MSG
ALLOC DD(SYSMDUMP) DSN(yourdsn) NEW SPACE(105) CYL
UNIT(unitname) VOLUME(volser) CATALOG
```

Following is a sample dump allocation for CA Roscoe users:

```
FREE  SYSMDUMP
DELETEDSN SYSMDUMP
ALLOC SYSMDUMP dsn=SYSDUMP
SPACE=(cyl, (50,10))
DISP=(NEW,catlg)
VOL=
UNIT=
```

You must print SYSMDUMPs using the AMDPRDMP utility.

Note: You can also use the DDALLOC command to allocate the SYSMDUMP DD to an existing DSN or a SYSOUT class.

EQUATE Command (Specify Symbolic Name)

Use the EQUATE command to equate a data item with a symbolic name. You may then refer to the data item by either the data item or the symbolic name. View the symbolic names using the LEQ command or delete by using the DROP command. The EQUATE command supports qualified data items. Equate without operands causes the Equate panel to display. For more information on the EQUATE command, see Breakpoint Panels in the chapter "Debugging Panels."

Syntax

Use the following syntax for EQUATE:

```
EQUATE <symbolic-name data-item>
```

You can enter this command in its abbreviated form: EQU.

Variables

Use the following variables with the EQUATE command:

symbolic-name A user-defined symbolic name for a data item

data-item An elementary data item or qualified data item

Data-item can be a qualified data item with 1 or 2 levels of qualification, as described in the following examples:

COBOL: The following command equates NAME-ONE OF EMP-NAME OF PAYROLL-REC to the symbolic name N1. You may now specify N1 as the data item for control commands such as LDI and SET.

EQUATE N1 NAME-ONE OF EMP-NAME OF PAYROLL-REC

This variable may also be specified by equating the following:

EQU NO NAME-ONE
EQU EN EMP-NAME
EQU PR PAYROLL-REC

NO OF EN OF PR used as the data item in a control command also refers to NAME-ONE OF EMP-NAME OF PAYROLL-REC.

PL/I: The following command equates PAYROLL_REC.EMP_NAME.NAME_ONE to the symbolic name N1. You may now specify N1 as the data item for control commands such as LDI and SET.

EQUATE N1 PAYROLL_REC.EMP_NAME.NAME-ONE

You may also specify this variable by equating the following:

EQU NO NAME_ONE
EQU EN EMP_NAME
EQU PR PAYROLL_REC

PR.EN.NO used as the data item in a control command also refers to PAYROLL_REC.EMP_NAME.NAME_ONE.

The data item may be a fixed or variably subscripted data item or indexed data item. The following commands equate a variably subscripted data item to the symbolic name W1.

COBOL:

```
EQ W1 WA-TI-3A(S1,S2,S3)
```

PL/I:

```
EQ W1 WA_TI_3A(S1,S2,S3)
```

The data item can also be a qualified subscripted or indexed data item, as in the following examples:

COBOL: This command equates the symbolic name W1 to the fourth value in the non-unique table WA-1A in the unique table WA-TABLE-1:

EQU W1 WA-1A OF WA-TABLE-1(4)

PL/I: This command equates the symbolic name W1 to the fourth value in the non-unique table WA_1A in the unique table WA_TABLE_1:

EQU W1 WA_TABLE_1.WA-1A(4)

EXSUM Command (Display Execution Summary)

The EXSUM command provides an execution summary display of the current program. The display shows the number of executable statements, the number of statements currently executed during the test, and the percentage of statements currently executed during the test.

Note: Statements executed when COUNTS OFF is in effect are not counted by the EXSUM command and will not get included in the summary.

Syntax

Use the following syntax for EXSUM:

```
EXSUM
```

You can enter this command in its abbreviated form: EXS.

The following panel is a sample Execution Summary Display:

```
---------------- CA InterTest Batch Execution Summary Display -----------------

PROGRAM ID:  INSTST

Number of executable statements: 21

Number of statements currently executed during this test: 21

Percentage of statements currently executed during this test: 100
```

Note: If COUNTS NOSHOW is in effect, no statements are counted and the Execution Summary Display will show N/A for Not Applicable for all three counts.

FILES Command (Display COBOL FD Status)

The FILES command displays the File Status panel. The File Status panel indicates whether the data sets in the FD section are open or closed, and it gives the DSORG and DCB information for the files. For more information on File Status Panels, see the chapter "Debugging Panels."

Syntax

Use the following syntax for FILES:

```
FILES
```

You can enter this command in its abbreviated form: FI.

Usage Notes

IMS databases are not supported on this display. This command is not supported for assembler or PL/I programs.

FM Command (Invoke CA File Master Plus)

The FM command invokes the CA File Master Plus product, if you have that product installed. For details on using CA File Master Plus, see the *CA File* Master Plus *User Guide*.

Syntax

Use the following syntax for FM:

```
FM
```

Note: This command is only valid when testing a program under ISPF.

FREQ Command (Controls FREQ Counter)

The frequency command controls the statement frequency counter, but the frequency command and the NOFREQ command control whether the counters are displayed.

Note: The FREQ command is replaced by the COUNTS command, which is described earlier in this chapter.

Syntax

Use the following syntax for FREQ:

```
FREQ <ON|OFF|RESET>
```

You can enter the FREQ command in its abbreviated form: FR.

Variables

Use the following variables with the FREQ command:

ON Enables frequency counting. This is the default. This command causes the Intercept panel to shift to the right six characters and displays one of the following in the left margin:

nnnn A four-digit integer indicating the number of times the statement has executed. If the counter exceeds 9999, the number is represented as follows:

*nnn*K = *nnn* x 1 thousand (for example, 010K= 10 thousand)

*nnn*M=*nnn* x 1 million (for example, 010M= 10 million)

*nnn*G=*nnn* x 1 billion (for example, 010G= 10 billion)

	If your frequency count has exceeded 9999 and you wish to see an exact count, issue an I (Info) line command to the left of the line. The full count is displayed in the upper right-hand corner of the Intercept panel.
--->	This line contains the beginning of a valid statement, but it has not yet been executed.
Spaces	The line number is not the beginning of a statement.
OFF	Disables frequency counting.
RESET	Zeroes the frequency counters for the currently qualified program.

Usage Notes

When you initially invoke the application, the frequency counter is turned on but is not displayed. To enable the display, enter the FREQ command without operands. Subsequently, if the FREQ OFF command is issued, the counters are no longer maintained, but the residual counts remain on the screen. To remove the counters and arrows from the screen, enter the NOFREQ command.

The status of the frequency command is maintained over sessions.

GO Command (Continue Test Session)

The GO command causes the test session to continue until the next intercept occurs.

Syntax

Use the following syntax for GO:

```
GO <Statement-number|paragraph-name|procedure-name|label-name>
```

Variables

Use the following variables with the GO command:

statement-number	Specifies a statement-number in the program where the program resumes execution.
paragraph-name	Specifies a paragraph-name in the program where execution resumes.
procedure-name	Specifies a procedure-name in the PL/I program where execution resumes.
label-name	Specifies a label-name in the assembler or PL/I program where execution resumes.

Specify a statement number, paragraph, procedure, or label name as the execution start point. GO with no operands resumes execution at the next logical statement unless the program was interrupted because of an ABEND. In this case, the program resumes execution at the statement where the ABEND occurred.

The program executes until one of the following occurs:

- A breakpoint is encountered.
- An ABEND occurs in the program.
- The step count (specified by the STEP command) is exhausted.
- The user interrupts the test by pressing the ATTENTION key.
- The program runs to normal completion.

Usage Notes

When using GO to modify the flow of execution of a COBOL program, be careful to avoid entering a paragraph and bypassing the PERFORM statement. When you reach the exit to the paragraph, you may receive a USER 519 ABEND because the references to the calling statement are not defined.

When using GO to modify the flow of execution of a PL/I program, be careful to avoid entering a block (that is, procedure, BEGIN block, or ON-unit) other than at the start of the block, or exiting a block other than at the END statement for the block, as it will cause unpredictable results.

INCLUDE Command (Execute CA InterTest Batch Commands)

The INCLUDE command reads and executes a set of application commands from a partitioned data set.

Syntax

Use the following syntax for INCLUDE:

```
INCLUDE member-name
```

Variables

Use the following variable with the INCLUDE command:

member-name	Specifies a valid member name of a partitioned data set (PDS) allocated to ddname INT1CLIB.

Usage Notes

Before you enter the INCLUDE command, perform an allocation to ddname INT1CLIB. Allocate the partitioned data set with an LRECL=80 and a RECFM=FB. Create the members using a text editor.

Use the INCLUDE command to execute a set of application commands for regression testing or retesting as code is modified during testing. For example, use the following commands to test a tax computation routine:

```
SET EMPLOYEE-NAME = 'PIERCE, MICHAEL'
SET EMPLOYEE-NUMBER = 3938
SET EMPLOYEE-STATE = 'CA'
WHEN TOOHIGH EMPLOYEE-SALARY GT 50000.00
AT 0100-CALC-FICA
```

Note: The INCLUDE command ignores all commands in the input stream following a command that causes the program to resume.

For example, GO or NEXT cause the program to resume execution; DISPLAY or LISTAT cause a transfer of panels. If the input stream includes these commands, any command following them is ignored.

LDA (or AUTOKEEP) Command (List Data Automatic)

The LDA or AUTOKEEP (list data automatic) command displays in the keep window all the data items referenced by the current statement.

The keep window appears on the Intercept panel immediately above the first line of the program listing.

Syntax

Use the following syntax for LDA or AUTOKEEP:

```
LDA <ON|OFF>
```

You may enter this command in its abbreviated form: AUTOK.

Variables

Use the following variables with LDA or AUTOKEEP:

ON	Enables the automatic list data facility.
OFF	Disables the automatic list data facility.

Usage Notes

LDA does not inhibit your ability to use the LDI (KEEP) or LDIX (KEEP HEX) commands to add static entries to the keep window. While automatically added data items are added and removed at each breakpoint, a static data item continues to be displayed in the keep window until removed manually.

When AUTOKEEP is in effect for an assembler routine, the operands for the current instruction are added to the keep window in register notation.

LDI (or KEEP) Command (List Data Item)

The list data item (LDI) command inserts a line that shows the data item and its value before the first line of the program listing on the Intercept panel. The data item appears on all subsequent Intercept panels until it is reset by the RDI (reset data item) command.

Syntax

Use the following syntax for LDI or KEEP:

```
LDI data-item|expression
```

Note: Data for variables are case-sensitive.

You can enter this command in its abbreviated form: K.

Variables

Use the following variables with LDA or AUTOKEEP:

data-item	An elementary data item or group name from the test program.
expression	For assembler programs, enter an expression in register notation.

Usage Notes

Modify elementary items from within the keep window. Type over the value displayed on the screen with the new value.

The LDI command supports group level items. To view the members of a data group, specify the group name as the data-item for the LDI command.

Use the LDI command to view COBOL SPECIAL REGISTERS. For example, this command places the current value of return code on the Intercept panel:

```
LDI RETURN-CODE
```

The data-item parameter can be a qualified data item, as in the following examples. Specify either 1 or 2 levels of qualification.

COBOL: Use this command to refer to the EMPLOYEE-NO in the MASTER-RECORD if the EMPLOYEE-NO is not unique.

LDI EMPLOYEE-NO IN MASTER-RECORD

PL/I: Use this command to refer to the EMPLOYEE_NO in the MASTER_RECORD if the EMPLOYEE_NO is not unique.

LDI MASTER_RECORD.EMPLOYEE_NO

The data-item parameter can be a fixed or variably subscripted data item or an indexed data item. For example, the following commands show the use of fixed subscripts:

COBOL:

```
LDI WA-TI-3A(1,2,3)
```

PL/I:

```
LDI WA_TI_3A(1,2,3
```

You can specify also variable subscripts, as follows:

COBOL:

```
LDI WA-TI-3A(SUB1,SUB2,SUB3)
```

PL/I:

```
LDI WA_TI_3A(SUB1,SUB2,SUB3)
```

In this case, the value displayed changes as the subscripted variables change. The values of the subscripts are also displayed. An * INVALID * message is placed next to any of the subscripts that are out of range.

The data-item can also be a qualified, fixed or variably subscripted data-item, or an indexed data item, as in the following examples:

COBOL: This command displays the fourth element of the Table WA-1A in WA-TABLE-1:

LDI WA-1A OF WA-TABLE-1 (4)

PL/I: This command displays the fourth element of the Table WA_1A in WA_TABLE_1:

LDI WA_TABLE_1.WA_1A (4)

For PL/I programs, the data-item parameter can be a locator qualified based variable. For example, use the following command to display the allocation of ELEMENT that is based off of NEXT_ELEM_PTR.

```
LDI NEXT_ELEM_PTR->ELEMENT
```

Both the data-item and the locator qualifier can also be qualified and fixed or variably subscripted. CA supports up to 15 levels of locator qualification.

Special Registers

Special registers are compiler-generated storage areas whose primary use is to store information produced through one of COBOL's specific features.

You may display the value of the following special registers in the application:

```
RETURN-CODE
SORT-RETURN
SORT-FILE-SIZE
SORT-CORE-SIZE
SORT-MODE-SIZE
LABEL-RETURN
```

For more information regarding these special registers, refer to your *IBM VS COBOL for* OS/VS *Manual*, *IBM COBOL II Application Programming Language Reference Manual,* or *COBOL/370 Language Reference Manual*.

LDX (or KEEPX) Command (List Data Item in Hex)

The list data item in hexadecimal (LDX) command inserts three lines showing the data item and its value before the first display lines of the program listing on the Intercept panel. The second and third lines display the data representation in hex format showing zones and numerics, respectively. The data item appears on all following Intercept panels until it is reset by the RDI (reset data item) command.

Syntax

Use the following variables with LDX or KEEPX:

LDX *data-item*

You can enter this command in its abbreviated form: KX..

Variables

Use the following variable with the LDX or KEEPX commands:

data-item An elementary data item or group item from the test program.

Usage Notes

Modify elementary items from within the keep window. Type over the value displayed on the screen with the new value. Note that you cannot modify the zones and numerics portion of the data display.

The LDX command supports group level items. To view the members of a data group, enter the group name as the data-item for the LDX command.

Use the LDX command to view COBOL special registers. For example, use the following command:

```
LDX RETURN-CODE
```

The data-item parameter can be a qualified data item, as in the following examples. Specify either 1 or 2 levels of qualification.

COBOL: Use this command to refer to the EMPLOYEE-NO in the MASTER-RECORD if the EMPLOYEE-NO is not unique:

- LDX EMPLOYEE-NO IN MASTER-RECORD

PL/I: Use this command to refer to the EMPLOYEE_NO in the MASTER_RECORD if the EMPLOYEE_NO is not unique:

- LDX MASTER_REC.EMPLOYEE_NO

The data-item parameter can be a fixed or variably subscripted data item or an indexed data item. For example, the following commands show the use of fixed subscripts:

COBOL:

```
LDX WA-TI-3A(1,2,3)
```

PL/I:

```
LDX WA_TI_3A(1,2,3)
```

You can specify also variable subscripts, as follows:

COBOL:

```
LDX WA-TI-3A(SUB1,SUB2,SUB3)
```

PL/I:

```
LDX WA_TI_3A(SUB1,SUB2,SUB3)
```

In this case, the value displayed changes as the subscript variables change. The values of the subscripts are also displayed. An * INVALID * message is placed next to any of the subscripts that are out of range.

The data-item can be a qualified fixed or variably subscripted data item or an indexed data item, as in the following examples:

COBOL: This command displays the fourth element of the table WA-1A in WA-TABLE-1:

- LDX WA-1A OF WA-TABLE-1 (4)

PL/I: This command displays the fourth element of the table WA_1A in WA_TABLE_1:

- LDX WA_TABLE_1.WA_1A (4)

For PL/I programs, the data-item parameter can be a locator qualified based variable. For example, use the following command to display the allocation of ELEMENT that is based off of NEXT_ELEM_PTR:

```
LDX NEXT_ELEM_PTR->ELEMENT
```

Both the data-item and the locator qualifier can also be qualified and fixed or variably subscripted. CA supports up to 15 levels of locator qualification.

Special Registers

See the previous section, LDI (or KEEP) Command (List Data Item), for a description of the supported special registers.

LEQ Command (List Equated Names)

The LEQ command lists the equated symbolic names and data items set by the EQUATE command. Delete the symbolic names by using the DROP command.

Syntax

Use the following syntax with LEQ:

```
LEQ
```

LINKAGE Command (Format the COBOL Linkage Section)

The LINKAGE command displays the Linkage Display panel. Modify the data values on this display.

Syntax

Use the following syntax with LINKAGE:

```
LINKAGE   <data-item> <PRINT>
```

You can enter this command in its abbreviated form: LI.

Variables

Use the following variables with the LINKAGE command:

data-item	Causes a display of the data item, any subordinate data items, and their data values.
PRINT	Causes the data displayed by the command to be written in the session log.

Usage Notes

Consider the following LINKAGE SECTION definition:

```
LINKAGE SECTION
01 PARM-VALUE
    05 PARM-COUNT                PIC S9(4) COMP SYNC.
    05 PARM-DATA
        10 PARM-DATE             PIC X(5).
        10 PARM-TIME.
            15 PARM-HH            PIC XX
            15 PARM-MM            PIC XX
            15 PARM-SS            PIC XX
```

If you enter the LINKAGE command without operands, it formats and displays all of the preceding data items and their data values.

The command LI PARM-DATA formats and displays PARM-DATA, PARM-DATE, PARM-TIME, PARM-HH, PARM-MM, and PARM-SS.

The command LI PARM-SS formats and displays PARM-SS.

Important! The LINKAGE command should not be used until the COBOL program has executed the PROCEDURE DIVISION or ENTRY statement.

The LINKAGE command is not valid for assembler or PL/I programs.

If you are navigating the trace table and data monitoring was in effect for the currently highlighted statement, the values displayed for your data items are from the time before that statement was executed.

LISTAT Command (List Breakpoints)

The LISTAT command displays the Breakpoint Status panel. This is a listing of the unconditional breakpoints set in the currently qualified program. You can delete breakpoints from this panel as well by placing an **X** in the command column of the line the breakpoint to be deleted is listed on, and pressing enter.

Syntax

Use the following syntax with LISTAT:

```
LISTAT
```

LISTBP Command (List All Breakpoints)

The LISTBP command displays the Breakpoints panel. This is a listing of all conditional (WHEN) and unconditional (AT) breakpoints set in the application. You may also delete breakpoints from this panel by placing an **X** in the command column of the line the breakpoint to be deleted is listed on, and pressing Enter.

Syntax

Use the following syntax with LISTBP:

```
LISTBP
```

You can enter this command in its abbreviated form: LBP.

LISTLBL Command (List Labels)

The LISTLBL command lists the labels that were added with the .label command.

Syntax

Use the following syntax with LISTLBL:

```
LISTLBL
```

LISTUSE Command (List Usings)

The LISTUSE command displays the Using Status panel. This is a listing of all usings set by the USING command. This command is only valid when debugging an assembler program.

Syntax

Use the following syntax with LISTUSE:

```
LISTUSE
```

You can enter this command in its abbreviated form: LISTU.

LISTWHEN Command (List WHEN Conditions)

The LISTWHEN command displays the When Conditions Display panel. This is a listing of the conditional breakpoints set in the currently qualified program. It lists both global and local conditional breakpoints. You may delete breakpoints from this panel as well by placing an **X** in the command column of the line the breakpoint to be deleted is listed on, and pressing Enter.

Syntax

Use the following syntax with LISTWHEN:

```
LISTWHEN
```

You can enter this command in its abbreviated form: LISTW.

MAP Command (Region MAP Display)

The MAP command invokes the Region Map Display Panel. This panel lets you display information about programs in storage. For more information, see the chapter "Core and the Map Options."

Syntax

Use the following syntax with MAP:

```
MAP
```

NEXT Command (Executing Verbs)

The NEXT command executes the next verb in the program. You will see the Next Intercept panel after entering NEXT.

Syntax

Use the following syntax with NEXT:

```
NEXT <nnn|P|BEFORE|AFTER|?|OVER|RETURN>
```

You can enter this command in its abbreviated form: N.

Variables

Use the following variables with the NEXT command:

nnn	Specifies the number of statements to execute. The default is one.
P	Specifies that all statements in the current paragraph that are not to be skipped will be executed, and execution stops at the next paragraph.
BEFORE	Specifies that when the NEXT command is used, execution stops before the statements are executed. Note this is the default.
AFTER	Specifies that when the NEXT command is used, execution stops after the statements are executed. **Notes:** If NEXT is set to AFTER, statements that cause execution to leave the current program, such as GOBACK for COBOL, will have no effect on the NEXT command. If data monitoring was in effect, all data item changes are captured at a point in time *before* COBOL statements referencing them are executed even when NEXT AFTER is set.
?	Displays a short message, reporting on whether NEXT stops before or after the specified number of statements.
OVER	Specifies that when the NEXT command is used executes through a verb, instruction, or statement and return to that line. This is to avoid breakpoints.
RETURN	Specifies that when the NEXT command is used to continue execution until the next CALL or PERFORM verb. Execution will stop when it hits a CALL or PERFORM execution.

When stopped on a PERFORM or CALL, the **NEXT OVER** command causes the test session to continue until the statement following the PERFORM or CALL is encountered, where a break point occurs.

When stopped within a paragraph or subroutine, the **NEXT RETURN** command causes the test session to continue until the session encounters the statement following the PERFORM or CALL that invoked the current paragraph or subroutine, where a break point occurs.

NOFREQ Command (Remove Frequency Counters)

Use the NOFREQ command to remove the frequency counters and arrows from the Intercept panel.

Syntax

Use the following syntax for NOFREQ:

```
NOFREQ
```

Note: This is used along with the FREQ command. This command and the FREQ command have been replaced by the COUNTS command.

OFF (or OFFU) Command (Remove Unconditional Breakpoints)

Use the OFF command to delete unconditional breakpoints set by the AT or UNCOND command, the A or U line command, the SKIP command, the S line command, or the Breakpoint panel.

If you specify the OFF command with no operands, the Breakpoint panel is displayed.

Delete a breakpoint by specifying OFF with the statement number where the breakpoint is set in the COMMAND field of the Intercept panel.

Syntax

Use the following syntax for OFF and OFFU:

```
 OFF
<statement-number|paragraph-name|procedure-name|label-name|ENTRY|EXIT|LABEL|DBCAL
L|ALL ENTRY|ALL EXIT|ALL LABEL|ALL DBCALL <(subcommand-list)>>
```

You can enter this command in its abbreviated form: O.

Variables

Use the following variables with the OFF or OFFU commands:

Statement-number	A program statement number where an existing breakpoint is deleted.
Paragraph-name	A COBOL paragraph name where an existing breakpoint is deleted.
Procedure-name	A PL/I procedure name where an existing breakpoint is deleted.
Label-name	An assembler or PL/I label name where an existing breakpoint is deleted.
ALL	Indicates all unconditional breakpoints be deleted.
ENTRY	Deletes the breakpoint set by the ENTRY operand of the AT or UNCOND command in the currently qualified program.
EXIT	Deletes the breakpoint set by the EXIT operand of the AT or UNCOND command in the currently qualified program.
LABEL	Deletes the breakpoint set by the LABEL operand of the AT or UNCOND command in the currently qualified program.
DBCALL	Deletes the breakpoint set by the DBCALL operand of the AT or UNCOND command in the currently qualified program.
ALL ENTRY	Deletes the breakpoints set by the ALL ENTRY operand of the AT or UNCOND command.
ALL EXIT	Deletes the breakpoints set by the ALL EXIT operand of the AT or UNCOND command.
ALL LABEL	Deletes the breakpoints set by the ALL LABEL operand of the AT or UNCOND command.
ALL DBCALL	Deletes the breakpoints set by the ALL DBCALL operand of the AT or UNCOND command.

Usage Notes

Set a breakpoint by paragraph, procedure, or label name, and delete by the statement number on which the paragraph is declared and vice versa. For example, consider the following code:

```
041400    0000-INITIALIZATION SECTION.
041500
041600      MOVE +20 TO VARIABLE-LENGTH-1.
041700      MOVE +25 TO VARIABLE-LENGTH-2.
041800      MOVE +30 TO VARIABLE-LENGTH-3.
```

Set a breakpoint on the first statement of the example by either of the following UNCOND commands:

```
UNCOND 41400
    or
UNCOND 0000-INITIALIZATION
```

The breakpoint could be removed by either of the following OFF commands:

```
OFF 41400
    or
OFF 0000-INITIALIZATION
```

OFFWN (or OFFC) Command (Remove Conditional Breakpoints)

Use the OFFWN command to delete conditional breakpoints set by the C, W or V line commands, by the WHEN command, or in the When panel.

If you specify the OFFWN command with no operands, the When panel displays.

Delete a breakpoint by specifying OFFWN and when-name of the breakpoint to be deleted in the Command field of the Intercept panel.

Syntax

Use the following syntax for OFFWN and OFFC:

```
OFFWN <when-name|ALL>
```

Variables

Use the following variables with the OFFWN or OFFC commands:

when-name A user-defined unique label specified in the WHEN command and used by the OFFWN command. When-name may be from one to eight characters in length. A when-name can also be a statement number. Variable change breakpoints set by the V line command generate their own when-names (see the description of the V line command earlier in this chapter).

ALL Indicates that all conditional breakpoints should be reset.

POINT Command (Direct PREV and ADVANCE Pointer)

The POINT command causes the PREV and ADVANCE commands to operate from the statement number, paragraph, or procedure, or label name designated in the POINT command.

Syntax

Use the following syntax for POINT:

```
POINT statement-number|paragraph-name|procedure-name|label-name
```

Enter this command in its abbreviated form: PO.

Variables

Use the following variables with the POINT command:

statement-number	A program statement number from which PREV and ADVANCE are used.
paragraph-name	A COBOL paragraph name from which PREV and ADVANCE are used.
procedure-name	A PL/I procedure name from which PREV and ADVANCE are used.
label-name	An assembler or PL/I label name from which PREV and ADVANCE are used.

Usage Notes

You must turn on the trace facility for the POINT command to function. For example, if the program abended on statement 13100 and you wanted to review the statements that were executed earlier, the POINT command could be used to establish a statement from which to use the PREV and ADVANCE commands. Suppose a part of the trace table contained the following:

```
13100
13000
  .
  .
  .
11000
10900
10800
10700
10300
```

```
10200
10100
09900
```

If you entered the statement PO 10300 in the command line, ADVANCE would display and highlight 10700, but PREV would display and highlight 10200.

If data monitoring was in effect when the statement that is highlighted from the POINT command was executed, the values displayed for your data items are from the time before that statement was executed.

PREV Command (Trace in Reverse)

The PREV command causes the display to scroll to the previously executed statement and highlights it. Use the PREV command to step backward through the execution of the program.

Syntax

Use the following syntax for PREV:

PREV <*nnn*|P nn>

You can enter this command in its abbreviated form: P.

Variables

Use the following variables with the PREV command:

nnn	Specifies the number of previously executed statements for the application to scroll. The default is one.
P	Specifies that the previously executed paragraph, procedure, or label is displayed.
nn	Specifies the number of previously executed paragraphs, procedures, or labels for the application to scroll. The default is one.

Usage Notes

You must turn on the trace facility for the PREV command to function. You turn on the trace facility by default when you first invoke the application. For further information on turning the trace facility on and off, see the TRACE command later in this chapter.

The PREV command visually displays the flow of execution of the program in reverse. It does not actually reverse the execution or execute the program in reverse.

The PREV command does not affect the point at which the application resumes execution.

If data monitoring was in effect when the statement that is highlighted from the PREV command was executed, the values displayed for your data items are from the time before that statement was executed.

PS Command (Print Stream)

The Print Stream (PS) command causes the currently displayed stream to be written to the session log for the current session.

Syntax

Use the following syntax for PS:

```
PS
```

Usage Notes

The PS command writes the entire stream, not just the portion displayed on the screen. For example, if only the data that is currently shown on the screen display is needed, CA Roscoe users should allocate an INT1PRNT DD data set and use the PRINT PF Key function. Native TSO users can use this command as well. ISPF users should use the ISPF PRINT facility.

QUALIFY Command (Set Current Program Id)

The QUALIFY command identifies the program to which all subsequent commands apply. This qualification is automatically reset by another QUALIFY command, or resulting from the flow of the execution. When testing more than one program, the listing switches automatically to the qualified program.

Syntax

Use the following syntax for QUALIFY:

```
QUALIFY program-id
```

Variables

Use the following variables with the QUALIFY command:

program-id Specifies the PROTSYM member name of the program to which the following commands apply.

The program-id of the currently intercepted program is automatically qualified at each program intercept. The qualified program-id of the program is displayed in the upper left corner of the display.

Usage Notes

The QUALIFY command is useful when you are testing multiple programs. For example, you are testing programs A and B. The initial breakpoint occurs in A, but you want to set a breakpoint in B. The following command switches the listing file in the Intercept panel to program B:

```
QUALIFY B
```

Use display commands to find and scroll to the statement where you want to set the breakpoint. Issue the AT command, and the breakpoint is set in program B. Issue the following commands:

```
QUALIFY A

GO
```

Your test resumes execution until the breakpoint in program B is encountered.

QUIT Command (Terminate Session)

The QUIT command stops the test session. The Execution Control panel displays so you can do additional testing. Pressing the END key twice in a row is the same as entering the QUIT command.

Syntax

Use the following syntax for QUIT:

```
QUIT
```

Usage Notes

The QUIT command is the same as END and CANCEL.

The first time you press the END key, the message USE "END" TO TERMINATE appears at the top of the screen. Press the END key again.

RDI or REMOVE Command (Reset Data Item)

The RDI command removes from the Intercept panel the lines placed there by the LDI (list data item) or the LDX (list data item in hex) commands, or the L or H line commands, and moves the program listing up.

Syntax

Use the following syntax for RDI or REMOVE:

```
RDI   data-item|ALL
```

You can enter this command in its abbreviated form: RE.

Variables

Use the following variables with the RDI or REMOVE command:

data-item Must be a data item in a LDI command.

ALL Specifies that all LDI items should be deleted from the Intercept panel.

RECORDS Command (Show COBOL Record Formats)

The RECORDS command displays the COBOL File Section Display panel. The File Section Display panel shows the record formats under the file descriptions (FDs). You can modify data values on this display.

Syntax

Use the following syntax for RECORDS:

```
RECORDS <data-item> <PRINT>
```

You can enter this command in its abbreviated form: REC.

Variables

Use the following variables with the RECORDS command:

data-item Causes a display of the data item, any subordinate level data items, and their data values.

PRINT Causes the data displayed by the command to be written in the session log.

Usage Notes

Consider the following COBOL RECORD definition:

```
FILE SECTION.
FD INPUT-FILE.
     BLOCK CONTAINS 0 RECORDS
     RECORDING MODE IS F.
01 INPUT-RECORD.
     05 INPUT-NUMBER              PIC X(4).
     05 INPUT-NAME                PIC X(36).
     05 INPUT-SSN.
        10 INPUT-SSN-XXX          PIC X(3).
        10 INPUT-SSN-YY           PIC X(2).
        10 INPUT-SSN-ZZZZ         PIC X(4).
```

If you enter the RECORDS command without operands, it formats and displays all of the previous data items and their values.

The command REC INPUT-SSN formats and displays INPUT-SSN, INPUT-SSN-XXX, INPUT-SSN-YY and INPUT-SSN-ZZZZ.

The command REC INPUT-SSN-ZZZZ formats and displays INPUT-SSN-ZZZZ.

The INPUT-FILE must be open to modify data with the RECORDS command.

The RECORDS command is not valid for assembler or PL/I.

If you are navigating the trace table and data monitoring was in effect for the currently highlighted statement, the values displayed for your data items will be from the time before that statement was executed.

REFRESH Command

Use the REFRESH subcommand when you have split the screen during test execution, and have used one screen to change the value of a data item that is currently displayed on the other screen (screen 2).

REFRESH, typed on screen 2, lets the application pick up and display the newly updated value.

Syntax

Use the following syntax for REFRESH:

```
REFRESH
```

REGS Command

The REGS command displays the current application's general registers. This command is only valid for assembler programs.

Syntax

Use the following syntax for REGS:

```
REGS <ON|OFF|DISPLAY|FLOAT>
```

You can enter this command in its abbreviated form: REG.

Variables

Use the following variables with the REGS command:

ON	Displays the general registers in the Keep window.
OFF	Removes the general registers from the Keep window.
DISPLAY	Displays a panel where the general registers, their high halfs, and the access registers are displayed.
FLOAT	Displays a panel where the floating point registers are displayed.

RESET Command

RESET removes column numbers that are displayed on the Intercept or Display panel by the COLUMN command.

Syntax

Use the following syntax for RESET:

```
RESET
```

Usage Notes

The RESET command runs from any panel that has a command line that accepts the COLUMN command.

RESTART Command

The RESTART command re-initiates a batch link job to the beginning of the current step being debugged.

```
sRESTART
```

Usage Notes

This command is only valid under batch link. Foreground users should use the QUIT command.

RUN Command (Resume Execution)

The RUN command resumes execution of the program ignoring all breakpoints. The program continues to execute until normal termination or until it abends. The Execution Control panel displays.

This command removes any capability for further program debugging. The application will *not* intercept abends.

However, if an optional statement number is passed to the RUN command, when execution from the RUN command hits that statement, the RUN command is converted into a GO command. This means that existing breakpoints are honored again.

Note: Using the RUN command during a batch link session also issues a SUSPEND command, releasing the terminal.

Syntax

Use the following syntax for RUN:

```
RUN <statement-number>
```

Variables

Use the following variables with the RUN command:

Statement-number	A program statement number where the RUN command is converted to a GO command.

SDWA Command (System Diagnostic Work Area)

The SDWA command displays the System Diagnostic Work Area if a failure occurs. The panel displayed gives the following information: the SDWA address, the ABEND code, the PSW, the program name, the EP address, the offset, and the contents of the registers at the time of failure.

Syntax

Use the following syntax for SDWA:

```
SDWA
```

SET Command (Change Data Item Value)

Use the SET command to change the value of a data item.

Syntax

Use the following syntax for SET

```
SET data-item <=> value|LOW VALUES|HIGH VALUES|SPACES|ZERO|ZEROES|X'hex char string'
```

Note: Data for variables are case-sensitive.

Use the following variables with the SET command:

data-item	A data item in the monitored program or a COBOL special register. Data-item may be a fixed or variably subscripted data item or an indexed data item. It may also be a qualified data item.
value	The new value of the data item.
LOW-VALUES	Indicates that the value of the data item is the figurative COBOL constant of hexadecimal '00', which is the lowest value in the computer's collating sequence.
HIGH-VALUES	Indicates that the value of the data item is the figurative COBOL constant of hexadecimal 'FF', which is the highest value in the computer's collating sequence.
SPACES	Indicates that the value of the data item is one or more blanks or spaces.
ZEROS | ZEROES	Indicates that the value of the data item is one or more zeros.
hex-char-string	1 to 20 valid hexadecimal characters. The data is replaced from left to right.

Usage Notes

Alter data by overtyping on any of the data displays (from DISPLAY, LINKAGE, and RECORDS command). Change elementary data items using the SET command on the Intercept panel.

```
SET EMPLOYEE-ID = '123-45-6789'
```

This entry sets the data item EMPLOYEE-ID to a value of 123-45-6789. Use this command with the command library facility to initialize areas of data for testing. See the INCLUDE command described earlier in this chapter for further information on the command library facility.

Modify fixed or variably subscripted data items or index data items, that is, table entries, using the SET Command, as in the following examples:

COBOL:

- SET WA-TI(4) = 6

Sets the fourth element in the table WA-T1 to 6.

- SET WA-TI2(SUB1,SUB2) = 'S1S2'

Sets an element in WA-T2 depending on the values of SUB1 and SUB2 to S1S2.

- SET WA-T1-1A OF WA-TABLE-1 (2)= 2

Sets the second element of the non-unique table WA-T1-1A in the unique table WA-TABLE-1 to 2.

PL/I:

- SET WA_TI(4) = 6

Sets the fourth element in the table WA_T1 to 6.

- SET WA_TI2(SUB1,SUB2) = 'S1S2'

Sets an element in WA_T2 depending on the values of SUB1 and SUB2 to S1S2.

- SET WA_TABLE_1.WA_T1_1A(2) = 2

Sets the second element of the non-unique table WA_T1_1A in the unique table WA_TABLE_1 to 2.

- SET WA_TBL_PTR->WA_T1_1A(2) = 2

Sets the second element of the based table WA_T1_1A in the table allocation pointed to by WA_TBL_PTR.

If you are navigating the trace table and data monitoring was in effect for the currently highlighted statement, the SET command is not enabled. You must first issue the CS command to reset the tracing and then issue the SET command.

SKIP Command (Skip a Statement)

The SKIP command causes the statement number specified to be skipped when the program is executed.

The Intercept panel shows an S to the left of any statements that are to be skipped.

Skip is considered an unconditional breakpoint and may be reset using the OFF control command or the O or X line command.

Syntax

Use the following syntax for SKIP:

```
SKIP statement-number
```

Variables

Use the following variable with the SKIP command:

statement-number	Specifies the statement-number in the program that is to be skipped when the program is executing.

Usage Notes

Skipping a statement causes the execution of that statement to be bypassed entirely, with execution resuming at the verb or label whose statement number is nearest to and greater than the statement being skipped. In some instances, this may not result in the required execution path, with execution resuming at a statement that would otherwise not have been executed.

This can occur when the statement being skipped is any of the following statements:

- The last statement in the program
- The last statement in a PERFORMed paragraph
- Immediately followed by an ELSE statement

Important! In those instances that may not result in the required execution path, you should avoid using a SKIP command, and instead use an unconditional breakpoint with an associated GO command.

To set an unconditional breakpoint, type AT on the command line and press Enter. When prompted, enter the statement number where the breakpoint should occur, which is the statement you wish to skip. Type **GO nnnnn** in the Command field, where *nnnnn* is the number of the statement at which execution should resume.

In this example, a SKIP command on statement 917 would cause execution to resume at statement 919, which is not the required execution path:

```
         000915             IF A NOT EQUAL ZERO

         000916                  MOVE A TO OPTION-CHOSEN
      S  000917                  PERFORM PROCESS-SELECTION
         000918             ELSE
         000919                  DISPLAY 'ERROR: INVALID SELECTION'
         000920                  MOVE 12 TO RETURN-CODE.
         000921             PERFORM LOG-RESULT.
```

Instead, use the AT command to force execution to resume at statement 921:

```
-------------------- CA InterTest Batch BREAKPOINT PANEL ---------------------
OPTION ===> S

   S - SET A BREAKPOINT
   R - RESET A BREAKPOINT
   L - OR BLANK, LIST THE BREAKPOINTS

SPECIFY PROGRAM NAME AND STATEMENT NUMBER:
   PROGRAM NAME ===>
   STATEMENT    ===> 917

   COUNT        ===>          AFTER ===> N  (Y/N)

COMMANDS TO BE EXECUTED AT BREAKPOINT:
===> GO 921
```

You should also be aware that the use of a SKIP command may result in incorrect results when the program has been optimized, either by the IBM optimize option or by CA Optimizer or CA Optimizer/II. Skipping a statement in an optimized program may result in the loss of calculations or intermediate results, which may affect future statements.

SLOW (or AUTOSTEP) Command (Slowly Resume Program Execution)

The SLOW command resumes execution of the program one verb at a time and displays the Intercept panel between the executions of each verb.

Syntax

Use the following syntax for SLOW or AUTOSTEP:

SLOW *<nnn|FAST>*

You can enter this command in its abbreviated form: AUTO.

Variables

Use the following variables with the SLOW command:

nnn Specifies the number of seconds to halt the display between verb executions. The default is two seconds.

FAST Executes the program without halting between screen refreshes.

Usage Notes

AUTOSTEP (or AUTO) is a synonym for SLOW.

Interrupt this feature by pressing the Attention key. For information on the Attention key, see Stopping a Looping Program in the chapter "Testing Procedures."

Occasionally when you press the Attention key, the program stops and the short message field SUBCOMMAND ABEND is displayed. If this occurs, issue the NEXT command to refresh the display.

SNAP Command (Produce a SNAP Report)

The SNAP command produces a CA SymDump Batch, CA Optimizer/II, or CA Optimizer SNAP report, if you have one of these products initialized. For details on the SNAP report, see the appropriate product's *User Guide*.

Syntax

Use the following syntax for SNAP:

```
SNAP
```

Note: Minimum release requirements are CA SymDump Batch 2.0, CA Optimizer/II 3.0, or CA Optimizer 7.0.

SPEED Command (Accelerate Program Testing)

The SPEED command deactivates the trace and frequency facilities.

Syntax

Use the following syntax for SPEED:

```
SPEED <ON|OFF>
```

Variables

Use the following variables with the SPEED command:

ON Turns off trace and frequency counting. Issues the TRACE OFF and the FREQ OFF commands.

OFF Enables tracing and statement frequency counting. Issues the TRACE ON and the FREQ ON commands.

Usage Notes

This feature is very useful when testing a large program. It turns off trace and frequency counting and therefore improves performance. For example, to test a COBOL program with 20,000 PROCEDURE DIVISION statements, set a breakpoint at statement 15000, and step through the program when you get to that point. The commands to perform this are as follows:

```
AT 15000
SPEED
```

When statement 15000 is reached, an Intercept panel is displayed. To step through the program from this point, enter the following to turn on trace and frequency counting:

```
SPEED OFF
STEP 1
```

STEP Command (Set Count for GO Command)

The STEP command sets the execution mode to step operation. For each GO command, the specified number of verbs is executed, and the STEP COUNT Intercept panel is displayed when that number of verbs has executed.

Syntax

Use the following syntax for STEP:

```
STEP <nnnn|BEFORE|AFTER>
```

You can enter this command in its abbreviated form: ST.

Variables

Use the following variable with the STEP command:

nnnn Specifies the number of verbs to be executed each time you enter a GO command. Setting the STEP count to 0 (zero) turns off step counting so that the next GO command executes until a breakpoint is encountered, an abend occurs, or the program reaches normal termination. The maximum step count that you can enter is 9999.

BEFORE Specifies that execution should stop before the statement that satisfies the STEP count. This is the default.

AFTER Specifies that execution should stop after the statement that satisfies the STEP count.

Notes:

If STEP is set to AFTER, statements that cause execution to leave the current program, such as GOBACK for COBOL, have no effect on the STEP count.

If data monitoring was in effect, all data item changes are captured at a point in time *before* COBOL statements referencing them are executed even when STEP AFTER is set.

Note: If no variable is used with the STEP command, the current STEP count and status is displayed in the upper right hand corner.

SUSPEND Command (Suspend a Batch Link Session)

Use the SUSPEND command to detach a terminal from a batch link debugging session without terminating the session.

Syntax

Use the following syntax for SUSPEND:

```
SUSPEND
```

You can enter this command in its abbreviated form: SUS.

Usage Notes

The SUSPEND command is only valid for batch link debugging sessions and you can only use it from the Monitor Control or Intercept panels.

A suspended batch link session becomes immediately available for selection from the Batch Link Selection panel.

Reconnect a suspended session by the same user or by a different user. Regardless of which user selects the suspended session, the session is resumed from exactly the same point at which the session was suspended, and continues to use the same profile member that was defined in the INT1OPTS file.

TRACE SOURCE Command (Trace with Source Code Displayed)

The TRACE SOURCE command functions similar to the TRACE command, but displays the code being traced as well as the statement number.

Syntax

Use the following syntax for TRACE SOURCE:

```
TRACE SOURCE n
```

You can enter this command in its abbreviated form: TR SO.

Variables

Use the following variables with the TRACE SOURCE command:

n Specifies the maximum number of trace entries to be displayed. The default is 500.

TRACE Command (Controls Program Trace Entries)

If the TRACE command does not have an operand, it displays the Program Trace Display panel. This panel shows statement numbers in the order they were executed.

Syntax

Use the following syntax for TRACE:

```
TRACE <ON|OFF|n>
```

You can enter this command in its abbreviated form: TR.

Variables

Use the following variables with the TRACE command:

ON Turns on statement tracing. The number of entries retained is set by the TRACE(n) parameter in the initialization PARMLIB member. The default is 500.

OFF Turns off statement tracing.

n Specifies the number of entries to be saved in the trace table and displayed by the TRACE command. The current trace table is cleared and reset. This overrides the value specified in the TRACE(n) initialization PARMLIB member.

Usage Notes

The trace setting is remembered from session to session.

It is possible to view the statements that have been executed in reverse by using the PREVIOUS command, instead of having to view the trace display.

If data monitoring is in effect for any program in your application, changing the trace table resets the effects of the DATAMON command.

TRP Command (Trace Paragraphs)

The TRP command invokes the Program Trace Display panel. This panel shows statement numbers and paragraph, procedure, or label names executed.

Syntax

Use the following syntax for TRP:

```
TRP
```

Usage Notes

The trace facility must be enabled for the TRP command to function.

USING Command (Assign a Register or Address to a DSECT or CSECT)

The USING command lets you assign a register or address for the debugger to use to map a DSECT or CSECT that has no current using in the program being debugged. This command is only valid when debugging an assembler program.

Syntax

Use the following syntax for USING:

```
USING dsect|csect register|address
```

Variables

Use the following variables with the USING command:

dsect	Assembler DSECT that you wish to map
csect	Assembler CSECT that you wish to map
register	Register that maps the DSECT or CSECT
address	Address that maps the DSECT or CSECT

WHEN (or COND) Command (Set Conditional Breakpoints)

The WHEN command sets conditional breakpoints. There are two types of conditional breakpoints and the format of the command syntax is different for each:

Use this Syntax format	For this Type of Conditional Breakpoint
1	Whenever the value of a specified data item changes
2	When a relationship between two data items is true, for example, 'SUBSCRIPT-1 GT 4'

Enter operands in the When panel or in the COMMAND field of the Intercept panel in one of the following formats:

Format 1	This type stops the program's execution any time the value of the specified data item changes. For example: WHEN whename1 R1498_STATUS.
Format 2	This type checks a relational expression for a TRUE condition before intercepting the program. For example: WHEN whename2 R1498_STATUS GT G

If the *when-name* specified is a statement number, the WHEN condition is only tested before executing that statement number. In this case, the application does not test for the condition before each statement. This local type of conditional breakpoint reduces computing overhead.

Notes:

- WHEN conditions that are set for subscripted or indexed items are checked only if the subscript or index is valid.
- If you use the WHEN command without operands, the WHEN panel is displayed. Set, reset, or list conditional breakpoints from this displayed WHEN panel.

Syntax Format 1: When the value of a specified data Item changes

Syntax

Use the following syntax for WHEN or COND:

```
WHEN <when-name data-item-1> <BEFORE|AFTER> <(subcommand-list)>
```

You can enter this command in one of its abbreviated forms: WH, WN or C.

Variables

Use the following variables with the WHEN or COND commands:

when-name	A user-defined unique label or a statement number; if the when-name is a statement number, the when condition is only checked before the statement is executed. It is also displayed in the title line of the W when-name Intercept panel.
data-item-1	The data item is examined before each verb is executed unless the when-name is a statement number in which case the data item is only examined before executing that statement (local conditional breakpoint). Whenever the value of the data item changes, a W when-name Intercept panel is displayed. Data-item-1 may be a fixed or variably subscripted, indexed, or qualified data item.
BEFORE	Specifies that execution should stop *before* the statement when the condition is met. (That is, the application detected that the value of the specified *data-item* has changed.) This is the default behavior.
AFTER	Specifies that execution should stop *after* the statement when the condition is met. (That is, the application detected that the value of the specified *data-item* has changed.)

subcommand-list A list of application commands, separated by semicolons, to be executed when the breakpoint is encountered during the test. The subcommand-list is enclosed in parenthesis.

Syntax

Use the following syntax for WHEN or COND:

```
WHEN  <when-name <data-item-1 <operator <data-item-2|value> <BEFORE}AFTER>>
       <(subcommand-list)>>>
```

You can enter this command in one of its abbreviated forms: WH, WN or C

Variables

Use the following variables with the WHEN or COND commands:

data-item-1 Based on the value of the operator, data-item-1 is compared to data-item-2. Whenever the expression is true, a W when-name Intercept panel is displayed. Data-item-1 may be a fixed or variably subscripted, indexed, or qualified data item.

operator A relational operator used to compare data-item-1 to data-item-2. Values can be any one of the following:

EQ	or =	Equal to
NE	or ?=	Not equal to
LE	or <=	Less than or equal to
GE	or >=	Greater than or equal to
LT	or <	Less than
GT	or >	Greater than

Note: => and =< are not permitted.

data-item-2 Compared to data-item-1. When the comparison is true, a W when-name Intercept panel is displayed. A value can be used in place of data-item-2. Data-item-2 may be fixed subscripted, an indexed data item, or a qualified data item.

value Specifies a constant value, which you can use in place of data-item-2. This value may be one of the following figurative COBOL constants: LOW-VALUES, HIGH-VALUES, ZEROS, or SPACES. This value may also be hex data in the format following:

X'hex-char-string'

where hex-char-string is 1 to 20 valid hexadecimal characters.

BEFORE	Specifies that execution should stop *before* the statement when the condition is met. (The specified *condition* is TRUE.) This is the default behavior.
AFTER	Specifies that execution should stop *after* the statement when the condition is met. (The specified *condition* is TRUE.)
subcommand-list	A list of application commands, separated by semicolons, to be executed when the breakpoint is encountered during the test. The subcommand-list is enclosed in parenthesis.

Usage Notes

Global conditional breakpoints (the when-name is not specified using a statement number) are breakpoints whose conditions are checked before every executed statement and are very resource intensive and may require considerable overhead. When possible, use local conditional breakpoints (the when-name is specified using a statement number) which are only checked at the time a particular statement is executed. See the Line Commands section earlier in this chapter for information on setting local conditional breakpoints with the line commands C, W, and V.

A *before* breakpoint is a breakpoint that stops *before* a statement is executed. A *before* breakpoint will never stop before a statement that *makes* the condition true. It would only stop before the next statement which follows the statement that makes the condition true. Thus, a *before* WHEN condition is evaluated before a statement is executed, and if true, the breakpoint occurs immediately, before that statement is executed.

An *after* breakpoint is a breakpoint that stops *after* a statement that makes the condition true. Thus, an *after* WHEN condition is evaluated after the statement is executed, and if true, the breakpoint occurs immediately, after the statement is executed.

* Command (Add Comments in Session Log)

The * command lets you specify a comment to be inserted in the session log. The text following the * is inserted in the current session log exactly as typed.

Syntax

Use the following syntax for *:

```
* text
```

Variables

Use the following variable with the /clist command:

Text Comment to be inserted in the session log

/clist Command (Execute CAMRCMD)

This product provides a facility for executing a CLIST from within the application. By typing in a slash (/) followed by a string, the CAMRCMD CLIST is called with a parameter, which is the string that follows the slash. Review CLIST CAMRCMD for further modification instructions.

Syntax

Use the following syntax for /clist:

```
/clist
```

Variables

Use the following variable with the /clist command:

Clist The parameter with which the CAMRCMD CLIST is called.

Report Commands

The software provides report commands to create reports that you can print and analyze offline. The reports show path coverage of program execution and a histogram report displaying the execution frequencies by statement number in graphic format. Both reports have an execution summary showing the number of executable statements, the number of statements executed, and the percentage of path coverage.

Note: The frequency counter must be on (using the COUNTS command) during testing if any of these reports are to be produced.

The reports use the ddname INT1REPT. The allocation of INT1REPT can be either to a SYSOUT class or to a disk data set. Sample allocations are as follows.

Allocation of INT1REPT to a SYSOUT class:

```
ALLOC DD(INT1REPT) SYSOUT(A)
```

Allocation of INT1REPT to a disk data set:

```
ALLOC DD(INT1REPT) DSN(INT.REPORT) NEW SPACE(2 1) CYL
```

If the ddname is allocated to a disk data set, the issuance of each report command completely rewrites the data set. If you would like to obtain both reports, perform the following steps:

1. Allocate the ddname INT1REPT to a data set.
2. Enter the HIST command.
3. Reallocate the INT1REPT to another data set.
4. Enter the XSUM command.

HISTOGRAM Command (Graph Execution Frequencies)

The HISTOGRAM command writes an execution histogram to ddname INT1REPT. The histogram consists of one line per executable statement. The line contains the statement number, the execution frequency counter, and a graphic representation of the number of times the statement has been executed.

The final page of the report identifies the number of statements in the program, an indication of the number of statements that were executed, and the percentage of the statements that were executed during this test.

Syntax

Use the following syntax for HISTOGRAM:

```
HIST <scale>
```

You can enter this command in its abbreviated form: HI.

Use the scale parameter to specify the histogram scale to be used. For example, a scale value of 10 would cause one asterisk to be printed on the histogram for every 10 executions of a statement.

If you do not specify a scale, the application automatically selects a scale in a multiple of five that causes no overflow to occur.

The histogram produces correct output only if the test was produced with FREQ ON and TRACE ON. A sample histogram report follows:

```
1CA InterTest/Batch       Execution Histogram For Program-ID: CAMRCOB2     Date
0
 000875 0001 |*
 000876 0001 |*
 000886 0001 |*
 000887 0001 |*
 000889 0001 |*
 000890 0001 |*
 000891 0001 |*
 000895 0002 |**
 000896 0002 |**
 000897 ---- |
 000899 0002 |**
 000900 0002 |**
 000901 0001 |*
 000902 0001 |*
 000904 ---- |
 000906 0001 |*
 000908 ---- |
 000909 0002 |**
 000910 0002 |**
 000914 0001 |*
 000915 0001 |*
 000917 0001 |*
 000920 0001 |*
 000921 ---- |
 000923 ---- |
 000925 0001 |*
 000931 0001 |*
 000932 0001 |*
 000933 ---- |
 000934 ---- |
 000935 ---- |
 000938 ---- |
 000945 ---- |
 000946 ---- |
 000953 ---- |
 000954 ---- |
 000957 ---- |
 000958 ---- |
```

A sample histogram report summary follows:

```
1CA InterTest/Batch       Execution Histogram For Program-ID: CAMRCOB2     Date
-Session Number: 1,890 For Userid user99
-Number of executable statements: 236
0Number of statements executed during this test: 78
0Percentage of statements executed during this test: 33
```

XSUM Command (Execution Summary Report)

The XSUM command writes an Execution Summary report to ddname INT1REPT. The summary report consists of the program listing with the execution frequency counter to the left of each valid executable statement.

The final page of the report identifies the number of executable statements in the program, an indication of the number of statements that were executed, and the percentage of the statements that were executed during this test.

Syntax

Use the following syntax for XSUM:

```
SUM <UNEXEC>
```

Variables

Use the following variable with the XSUM command:

UNEXEC

Specifies that the report should only contain statements that have not been executed.

A sample XSUM report follows:

```
PP 5740-CB1 RELEASE 2.3   JULY 24, 2014               IBM OS/VS COBOL

 16.20.10  DATE FEB   3,2014

        1                          16.20.10               FEB 3,2014

              100010  IDENTIFICATION DIVISION.
              100020  PROGRAM-ID.  SCRIPT.
              100030  DATE-COMPILED.  FEB 3,2014.
              100050    ENVIRONMENT DIVISION.
              100060    DATA DIVISION.
              100070    WORKING-STORAGE SECTION.
              100080    01  BIN-REC.
              100090     05  BIN1        PIC 9999 VALUE 0.
              100100     05  BIN2        PIC 9999 VALUE 0.
              100110     05  BIN3        PIC 9999 VALUE 0.
              100120     05  BIN4        PIC 9999 VALUE 0.
              100130     05  MOVE-DISP   PIC X(6).
              100140*
              100150    PROCEDURE DIVISION.
              100160*
0001          100170    INITIALIZATION SECTION.
0001          100180        DISPLAY 'PROGRAM STARTING'.
0001          100190    BIN SECTION.
0010          100200    LOOP-MOVE.
0010          100210        IF BIN1 > 8
              100220          OR BIN2 > 346
              100230          OR BIN3 > 77
              100240          OR BIN1 NOT = BIN4
              100250          OR MOVE-DISP = 'INTCOB'
0001          100260             MOVE ZERO TO BIN1
0001          100270             GO TO LOOP-MOVE-END.
0009          100280        ADD 1 TO BIN1.
0009          100290        ADD 3 TO BIN2.
0009          100300        ADD 9 TO BIN3.
0009          100310        ADD 1 TO BIN4.
0009          100320        MOVE BIN1 TO MOVE-DISP.
0009          100330        GO TO LOOP-MOVE.
0001          100340    LOOP-MOVE-END.
0001          100350        DISPLAY MOVE-DISP.
0001          100360        MOVE ZERO TO BIN3.
0022          100370    NEXT-LOOP.
0022          100380        IF BIN3 > 20
0001          100390           GO TO NEXT-LOOP-END.
0021          100400        ADD 1 TO BIN3.
0021          100410        GO TO NEXT-LOOP.
0001          100420    NEXT-LOOP-END.
0001          100430        DISPLAY MOVE-DISP.
0001          100440        MOVE ZERO TO BIN2.
0001          100450    LAST-LOOP.
0041          100460        IF BIN2 < 40
0040          100470          ADD 1 TO BIN2
0040          100480          GO TO LAST-LOOP.
0001          100490    LAST-LOOP-END.
0001          100500          DISPLAY BIN2.
0001          100510          DISPLAY 'END OF PROGRAM'.
0001          100520          STOP RUN.
```

A sample XSUM report summary follows:

```
CA InterTest Batch       EXECUTION SUMMARY REPORT FOR PROGRAM-ID: SCRIPT

DATE: 02/03/14 TIME: 16:21:07

SESSION NUMBER: 0127 FOR USERID USER01

NUMBER OF EXECUTABLE STATEMENTS: 000032

NUMBER OF STATEMENTS EXECUTED DURING THIS TEST: 000032

PERCENTAGE OF STATEMENTS EXECUTED DURING THIS TEST: 100
```

A sample XSUM UNEXEC report follows:

```
                    UNEXECUTED STATEMENTS AND PARAGRAPHS REPORT
--->  000457                    MOVE 'B' TO R1498-STATUS
--->  000464                                MOVE 'T' TO R1498-STATUS
--->  000468                            MOVE 'H' TO R1498-STATUS.
--->  000478                     PERFORM 900-WRITE-ERROR THRU
--->  000480                     GO TO RETURN-TO-OPSYS.
--->  000497                 GOBACK.
--->  000513                     GO TO DEMONSTRATE-SLOW-COMMAND.
--->  000515                     GO TO DEMONSTRATE-CONDITIONAL-BKPT.
--->  000517                     GO TO DEMONSTRATE-SPLIT-SCREEN.
--->  000519                     GO TO DEMONSTRATE-LOOP-DETECTION.
--->  000521                     GO TO DEMONSTRATE-PREV-COMMAND.
--->  000523                     GO TO DEMONSTRATE-PROCESS-TABLE.
--->  000525                     GO TO DEMONSTRATE-HIST-COMMAND.
--->  000531                DEMONSTRATE-SLOW-COMMAND.
--->  000532                    MOVE ZERO    TO OPTION-CHOSEN.
--->  000537                SLOW-COND.
--->  000538                    MOVE SPACES  TO X-AXIS.
--->  000539                    MOVE ZEROES  TO YTD, TOTAL-GROSS.
--->  000541                PAY-CALC.
--->  000542                    PERFORM POPULATE-GRAPH THRU POPULATE-GRAPH-EX
--->  000545                PAY-CALC-EX.
--->  000546                    MOVE SPACES      TO CHEQUE-LINE.
--->  000547                    MOVE TOTAL-GROSS TO CHEQUE-AMOUNT.
--->  000548                PAY-RETURN.
--->  000549                    GO TO OPTIONS.
--->  000551                POPULATE-GRAPH.
--->  000552                    MOVE MONTH-ITEM(SUB-4)     TO X-AXIS-YTD.
--->  000553                    MOVE TOKEN-ITEM            TO X-AXIS-R(SUB-4).
--->  000554                    ADD  MONTHLY-AMOUNT(SUB-4) TO YTD.
--->  000555                    MOVE YTD                   TO TOTAL-GROSS.
--->  000556                POPULATE-GRAPH-EX.
--->  000557                    EXIT.
--->  000558                DEMONSTRATE-CONDITIONAL-BKPT.
--->  000559                    MOVE ZERO    TO OPTION-CHOSEN.
--->  000561                CONDITIONAL-BREAKPOINT.
--->  000562                    MOVE ZEROES  TO BILL-YTD, SUB-TOTAL.
--->  000568                    PERFORM BILL-CALC THRU BILL-CALC-EX
--->  000572                    MOVE SPACES      TO CHEQUE-LINE.
--->  000573                    MOVE TOTAL-GROSS TO CHEQUE-AMOUNT.
--->  000574                    GO TO OPTIONS.
--->  000576                BILL-CALC.
--->  000577                    ADD BILLING-AMOUNT(SUB-7) TO SUB-TOTAL.
--->  000578                    MOVE SUB-TOTAL            TO BILL-YTD.
--->  000579                BILL-CALC-EX.
--->  000580                    EXIT.
--->  000583                DEMONSTRATE-SPLIT-SCREEN.
--->  000584                    MOVE ZERO    TO OPTION-CHOSEN.
--->  000589                SPLIT-SCREEN.
--->  000590                    MOVE  R1498-TIMEO TO R1498-TIME.
--->  000591                    GO TO OPTIONS.
--->  000600                DEMONSTRATE-PREV-COMMAND.
--->  000601                    MOVE ZERO    TO OPTION-CHOSEN.
```

Session Log Facility (Review Debugging Session)

The application session log facility records and subsequently enables a user to review the activities of a debugging session.

The session log contains the following:

- An entry identifying the user ID and session number
- Two lines for each program intercept. The first identifies the program and intercept reason code. The second is a copy of the TRACE line as it appeared on the Intercept panel.
- All input commands
- Any comments entered by the user using the * command on the Intercept panel during the testing session
- Any WORKING STORAGE SECTION, LINKAGE SECTION, or RECORDS data displayed using the DI, LI, or RE command with the PRINT option or the P line command

A sample session log with dynamic symbolic support deactivated follows:

```
*INTERTEST BATCH SESSION #1168 FOR USER USER01  BEGINS AT 12:59:30 ON 09/09/14.
*SYMBOLIC FOR CAMRASM LOADED FROM==> NDVRTS.USER01.PROTSYM
*  INTERCEPT IN PROGRAM CAMRASM  AT #000005 REASON: *INITIAL*
*
AT START
AT GOBACK
GO

*  INTERCEPT IN PROGRAM CAMRASM  AT #000025 REASON: UNCOND BEFORE
*000025
GO

*  INTERCEPT IN PROGRAM CAMRASM  AT #000147 REASON: ABEND S0C7
*000147 000145 000144 000142 000141 000140 000139 000137 000136 000135 000134
SET TASKNUM = 0
GO

*  INTERCEPT IN PROGRAM CAMRASM  AT #000265 REASON: UNCOND BEFORE
*000265 000260 000259 000258 000257 000255 000254 000253 000252 000251 000250
XSUM
GO
```

A sample session log with dynamic symbolic support activated follows:

```
*INTERTEST BATCH SESSION #1169 FOR USER USER01  BEGINS AT 13:10:30 ON 03/29/14.
* SYMBOLIC FOR: CAMRASM AUTO POPULATED TO: NDVRTS.USER01.PROTSYM
* SYMBOLIC FOR: CAMRASM LOADED FROM=> NDVRTS.USER01.PROTSYM
*  INTERCEPT IN PROGRAM CAMRASM  AT #000005 REASON: *INITIAL*
*
AT START
AT GOBACK
GO

*  INTERCEPT IN PROGRAM CAMRASM  AT #000025 REASON: UNCOND BEFORE
*000025
GO

*  INTERCEPT IN PROGRAM CAMRASM  AT #000147 REASON: ABEND S0C7
*000147 000145 000144 000142 000141 000140 000139 000137 000136 000135 000134
SET TASKNUM = 0
GO

*  INTERCEPT IN PROGRAM CAMRASM  AT #000265 REASON: UNCOND BEFORE
*000265 000260 000259 000258 000257 000255 000254 000253 000252 000251 000250
XSUM
GO
```

The application session log facility records and subsequently enables a user to review the activities of a debugging session as shown in the previous panel. The session log may be copied into a member of a partitioned data set and subsequently edited and used as the subject of an INCLUDE command.

Pre-allocating the ddname INT1CLOG activates the session log. Allocate the log to a disk file or to a SYSOUT class. In either case, there is no need to specify any DCB information as the application sets it to:

```
RECFM=FB,LRECL=80,BLKSIZE=6160
```

Because user ID keeps the session log, each user who wants to use the log facility must have a session log data set. The data set must be a sequential file. A sample CLIST, INT1CLOG, exists in the CAMRCLIB data set to create a sample session log data set.

Chapter 5: Testing Procedures

This chapter describes how to do the following procedures:

- Prepare a program for testing.
- Identify to the application the program to be tested.
- Start and stop program testing at selected statements.
- Test IMS applications.
- Test DB2 applications.

This section contains the following topics:

Guidelines for Novice Users

This chapter provides an overview and checklist for novice users of the Foreground option. The following sections assist you in preparing your programs for testing and using the application panels and commands to test your programs. For detailed information on the application panels used with the Foreground Option, see the chapter “Debugging Panels”, and for detailed information concerning the commands used with the Foreground Option, see the chapter "Debugging Commands."

Preparation for Test Execution

Use the following steps to compile a program to be tested under the control of the application and to begin foreground test execution.

If you are testing an IMS program, first read this section, then read Testing IMS Applications. Also, if you are testing a DB2 application, see Testing IBM DB2 Applications. If you are testing an application that uses any database, batch link is the suggested method for debugging.

1. Compile the program that you wish to test using the procedure set up by your installer. This procedure has a post-compile step that generates the symbolic information needed to debug your program. Alternatively, you can do the following:
   - Add the symbolic information using the Utilities panel if you have the listing available.
   - Take advantage of the dynamic symbolic support feature, if you have CA Endevor SCM r7 SP2 or later installed.
2. Log in to TSO and enter the TSO command PROFILE WTPMSG. This command causes all messages that normally appear in the batch job log (write-to-programmer messages) to be sent to the terminal.

   If you place the PROFILE WTPMSG in the application CLIST, you will not need to reenter it each time you invoke the application.
3. Invoke the application (ask you technical support personnel for instructions, if needed).

   If you are testing an IMS or DL/I program, also review the instructions given in the section Testing IMS Applications.

   If you are testing under CA Roscoe, also review the instructions in the section Testing Under CA Roscoe.
4. When the Primary Option Menu displays, select Option 1: Foreground.
5. Fill in the relevant information on the Execution Control panel. Enter the name of the main program of the application that you want to debug, the execution parameters, if there are any, and the STEPLIB to be used for this test. If your execution JCL was converted to ALIB format, you can specify that ALIB data set and member here. Then all the information, described previously will be extracted from the ALIB file and the DDs that are required for execution will also be allocated. If you do not have an ALIB for this application, you will need to allocate any required DDs through the TSO ALLOC command or through Option 3: Allocate off of the Primary Option Menu.

   If you are testing an IMS or DL/I program, also review the instructions given in the section Testing IMS Applications.

6. Fill in the name of the program that you wish to debug in the Monitored Programs section of the Monitor Control panel. The name of the program is the PROTSYM member name for the program to be tested. Also fill in the data set name of the Symbolic file, the PROTSYM that was used when the program was compiled and post-processed. When all the information has been filled in, press Enter.

   If any of the programs specified on the Monitor Control panel are Visual Age PL/I for OS/390 programs and you entered a **Y** in the Monitor PL/I field, the PL/I PROTSYM/Load Module Map panel displays. Fill in the load module names for any PL/I programs that you want to debug.

   If you are using CA Endevor SCM to manage your load module and listing libraries, you can activate the dynamic symbolic support feature (DSS) by designating a PROTSYM file as the receiver for DSS post processor-created symbolic files. This reduces your pre-testing setup time and ensures you will always debug with the correct symbolic file associated with the load module being monitored.

   The execution of the application begins.

7. If no errors are encountered, the Initial Intercept panel displays. This is a display of the source statements beginning with the first executable statement. The title line should read *INITIAL* Intercept.

8. The program is ready to execute the application. Enter the GO command in the Command field and press Enter, or press the PF key set to GO.

   The program executes until one of the following conditions is encountered:

   - An abend occurs: The Intercept panel is displayed, highlighting the statement where the abend was detected.
   - The program runs to normal completion. The Execution Control panel displays with the return code in the upper right corner indicating that the run has completed.

Execute a Program Until an Error Occurs

When an abend occurs, use application commands to correct the bad data and continue the test.

For example, consider the following COBOL statements:

```
041600    MOVE +20       TO VARIABLE-LENGTH-1.
041700    MOVE FIELD1    TO SALARY-TO-DATE.
041800    MOVE +30       TO VARIABLE-LENGTH-3.
```

If FIELD1 contains invalid numeric data such as spaces, an S0C7 occurs when an attempt is made to execute statement 041700. When the S0C7 occurs, the Intercept panel displays: the abend reason code appears on the title line and statement 041700 is highlighted.

To check the contents of FIELD1, use the DISPLAY command to format and display WORKING STORAGE. To change the contents of FIELD1, move the cursor under the data displayed at the right of the data item, and key in the new value for the data item.

Press the END key to return to the Intercept panel. Then issue the GO command to resume the test session.

Stop a Looping Program

Press the Attention key or PA1 key to stop a looping program. This causes the debugger to stop execution at the next executed statement. The Attention key works for terminals that are attached to the system through an SNA controller. The PA1 key works for terminals attached to the system using other means. To have the PA1 key work on most terminals, you must first press the RESET key.

If you cannot locate the Attention or RESET and PA1 keys, ask your technical support person for assistance.

Note: If you are attempting to break from a looping program and the PA1 key does not work, press RESET and then the PA1 key.

Trace a Program in Reverse

A program can be stepped backward from the point of an abend to determine the cause of the abend condition.

Use the PREV command to trace the program's execution in reverse. The PREV command does not affect the contents of the application's variables. It also does not change the location where the GO command will resume execution.

To view the statement that has been executed, enter P or PREV in the Command field of the Intercept panel and press Enter.

Stop a Program at a Particular Statement

Stop a program at selected statements by setting unconditional breakpoints at those statements. From the Intercept panel, locate the statement where the program will be stopped, then use the U line command or the UNCOND command to set the unconditional breakpoint. To access the Intercept panel, review the Preparation for Test Execution section.

For example, consider the following COBOL statements:

```
041400  0000-INITIALIZATION SECTION.
041500
041600       MOVE +20 TO VARIABLE-LENGTH-1.
041700       MOVE +25 TO VARIABLE-LENGTH-2.

041800       MOVE +30 TO VARIABLE-LENGTH-3.
```

The program is stopped at the first line of this code so that you can examine the value of the VARIABLE-LENGTH variables used in the next three lines.

Set the breakpoint by typing a U in the left margin where the breakpoint will be set. In this example, type a U at statement 41400 and press Enter. Type GO and press Enter.

Use the UNCOND command also to set the breakpoint. It is especially useful if you want to automatically execute commands at the breakpoint. Assume that VARIABLE-LENGTH-4 was a data-item that should have been initialized in this paragraph. The MOVE +45 TO VARIABLE-LENGTH-4 statement was omitted from the program source code. Add the statement by setting a breakpoint and specifying the SET command on the UNCOND command.

Set the breakpoint using either of the following UNCOND command formats:

```
UNCOND 41400 (SET VARIABLE-LENGTH-4 = +45)
    or
UNCOND 0000-INITIALIZATION (SET VARIABLE-LENGTH-4 = +45)
```

Press Enter to set the breakpoint and then issue the GO command to execute the program. When the program begins execution of statement 041400, the Intercept panel displays. The title line reads BREAKPOINT INTERCEPT and line 041400 is highlighted. The Breakpoint Intercept panel displays before the statement is executed. The assignment of +45 to VARIABLE-LENGTH-4 has been done automatically.

Stop at a Statement and Step by Verb

Step through a section of code by setting unconditional breakpoints at particular statements and then use the STEP and GO commands to execute each verb. For example, consider the following statements:

```
049400  DIVIDE CONV-YEAR BY 4
049500      GIVING CONV-LEAP-COUNT
049600      REMAINDER CONV-LEAP-REMAINDER.
049700  ADD +1 TO CONV-LEAP-COUNT.
049800  IF CONV-LEAP-REMAINDER IS EQUAL TO ZERO
049900      THEN
050000      SUBTRACT +1 FROM CONV-LEAP-COUNT.
050100
050200  IF CONV-DAY IS NOT EQUAL TO ZERO
050300      THEN
050400      SUBTRACT +1 FROM CONV-DAY.
050500
050600  COMPUTE CONV-TOD =
050700      (((((((CONV-YEAR * 365) + CONV-LEAP-COUNT + CONV-DAY)
050800      * 24) + CONV-TIME-HOUR) * 60) * CONV-TIME-MINUTE)
050900      * 60) + CONV-TIME-SST.
051000
```

Assume that one of the IF clauses in the previous example is not functioning as expected. Step through this section of the code by using the following command sequence:

UNCOND 49400	Sets a breakpoint at statement 049400, the DIVIDE verb.
GO or GO key	Executes the program until statement 049400 is reached; then the Intercept panel displays. The title line reads UNCOND BEFORE INTERCEPT. Statement 049400 is highlighted.
STEP 1	Executes (or steps) one verb each time you enter the GO command.
GO or GO key	Executes one verb and redisplays the Intercept panel. This occurs each time you enter GO, until the step count is turned off. This process lets you monitor the flow of control through the program.

Assume that statement 050600 has been reached and you want to execute the program until this loop is reentered without entering GO commands for each statement. To reset the program to normal execution, you must turn off the step count and enter the GO command once more:

STEP 0	Turns off the step count (sets it to zero).

GO or GO key	Executes the program normally until statement 049400 is reached again. The Intercept panel displays, the title line reads UNCOND BEFORE INTERCEPT. Statement 049400 is highlighted.

Note: As an alternative to using the STEP 1 command in conjunction with the GO command, you can use the NEXT command. This command executes up to the next verb.

Stop a Program When a Subscript Overflows

Many S0C4 abends are caused by a subscript exceeding its expected range (overflowing). This causes unrelated data in a program to be overlaid or garbled. The application can stop a program when a particular data item contains or exceeds a value or another data item. This is done by setting a conditional breakpoint using the WHEN command.

For example, assume that subscript SUBS1 is exceeding its range (1 through 30). To stop the program at the statement where SUBS1 exceeds 30, enter the following command:

```
WHEN SUBSCRIP SUBS1 GT 30
```

This command stops the program when SUBS1 is greater than 30. The character string SUBSCRIP in the command is the name assigned to the WHEN condition. It is displayed in the title of the W when-name Intercept panel whenever this condition is encountered; for example, the title of the Intercept panel reads:

```
WHEN SUBSCRIP BEFORE INTERCEPT
```

The when-name identifies which WHEN condition was encountered. The when-name can also be a statement number. For example, assume that SUBS1 is updated in statement 15200 and that SUBS1 exceeds its range.

Enter the WHEN condition as follows:

```
WHEN 15200 SUBS1 GT 30
```

This way the condition will only be checked when the specified statement number is being executed.

Test IMS Applications

If you are licensed for the IMS Option, you can use the product to test and debug either batch or online IMS applications. Test the IMS programs with the product the same way as other programs, but you need to remember that the IMS program is not the first program executed when an IMS application is invoked; the IMS program is attached by the IMS Region Controller (DFSRRC00). This occurs whether the application is a DB/DC application (MPP or BMP) or a DB application (batch).

Your product interfaces with the IBM BTS (Batch Terminal Simulator) program, which you use to simulate online applications.

Notes:

- The Region Controller is a module, DFSRRC00. It is not the IMS control region. The IMS control region is an MVS address space that runs the program DFSRRC00. The application runs in an MVS address space, TSO that can also run the program DFSRRC00. Neither TSO nor CA InterTest Batch runs in the IMS Control Region.
- The instructions in this chapter are for debugging an application under Foreground, Option 1 on the Primary Option Menu. The suggested method for testing IMS applications is using Option 5, Batch Link. For more information on batch link, see the chapter, "Batch Link Facility."
- A facility exists for debugging your online IMS application without the use of BTS. For more information on debugging IMS DC applications, see the chapter "Batch Link Facility."

IMS Batch Program Testing

The JCL for executing a batch IMS program has an EXEC statement that specifies the module DFSRRC00. The application program name is specified as a parameter to the DFSRRC00 module. To facilitate testing IMS batch applications with your product, create a CLIST to allocate IMS and CA InterTest Batch-related data sets. Ask your technical support person for assistance in creating the CLIST, if needed.

The CLIST allocations are as follows:

IMS-related:

```
DFSRESLB  IMS  IEFRDER  DFSVSAMP
```

Note: On the ddname DFSRESLB, the IMS RESLIB data set is concatenated to itself. This is necessary to prevent a dynamic allocation to the ddname. If the IMS RESLIB data set is not concatenated to itself under the ddname DFSRESLB, USER 0684 abends and other unpredictable user abends can occur.

CA InterTest Batch-related:

```
INT1PARM  INT1LOAD  INT1PNNL  INT1PROF  INT1MSGL INT1CLOG INT1CLIB INT1REPT
```

Note: For additional information on CA InterTest Batch-related CLISTS, see the appendix "Basic Foreground Demo Session."

Preparation for Test Execution - Batch Programs

Replace Step 5 in the section Preparation for Testing Execution with the following information:

- On the Execution Control panel, specify the name of the IMS RESLIB data set as the load library.
- The first program to be executed is always DFSRRC00.
- The execution parameters are the same as the parameters used in a batch execution.

Generally, the execution parameters are DLI,membername,psbname. If the application program module name and the psbname are the same, then the psbname parameter does not need to be specified.

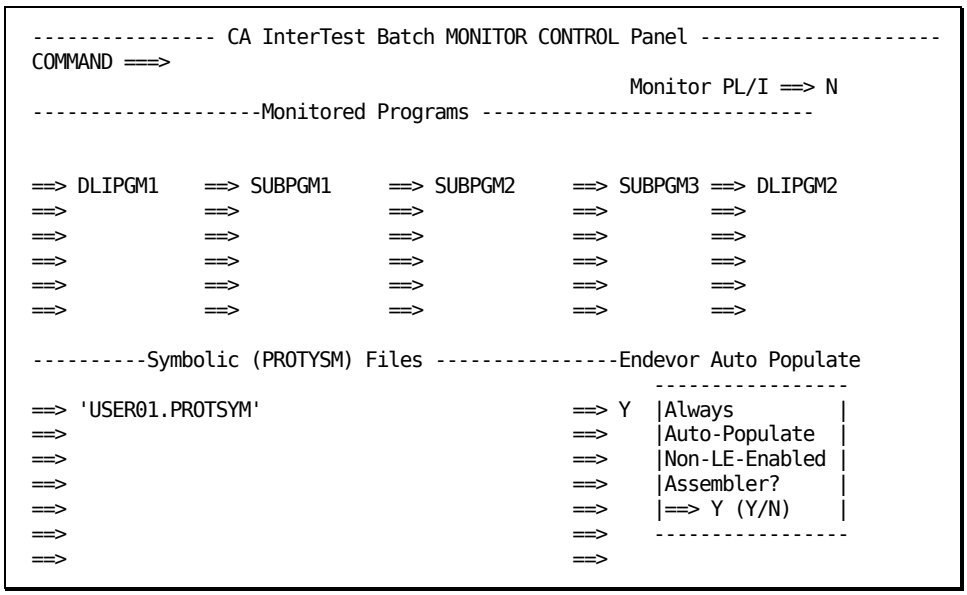
- The application requires a STEPLIB DD statement that identifies the load library containing the application program to be executed.
- You must also ALLOCATE the data bases that are used by the IMS program that will be tested.

The application generates a dynamic STEPLIB DD statement from the data set names listed under the STEPLIB portion of the Execution Control panel. For more information on dynamic STEPLIB facility in Execution Control Panel, see the chapter "Debugging Panels."

Use the Monitor Control panel to identify the programs that have their execution controlled by the application. Identify only the programs that you wish to debug with the application. Also, provide the PROTSYM files that contain the symbolic information for the programs that you would like to debug.

Example:

In this example assume that you have two driver modules to debug and test: DLIPGM1 and DLIPGM2. DLIPGM1 calls five subprograms: SUBPGM1, SUBPGM2, SUBPGM3, SUBPGMA, and SUBPGMB. DLIPGM2 calls SUBPGMA and SUBPGMB. SUBPGMA and SUBPGMB do not need to be debugged. In this run, they will be executed, but not symbolically debugged. Thus, their names do not need to be included on the Monitor Control panel.



Batch Procedures for BMP Testing

BMPs are batch IMS programs that run under the control of an online IMS control region. The job is executed in an IMS BMP region. The BMP accesses databases that are allocated to IMS. Thus, the CLIST you use to invoke the application will not include allocation statements for the databases.

If you need assistance in creating the CLIST, ask your technical support person.

Note: If you prematurely end your test session with an END, CANCEL, or QUIT command when testing BMPs, the BMP region does not know the testing session is ending. The BMP region may abend with an S13E.

The following screen shows you an example of an Execution Control panel for IMS BMP testing:

```
-------------- CA InterTest Batch EXECUTION CONTROL Panel ---------------------
COMMAND ===>

------------------------------ Allocations ----------------------------------

ALIB Dsname   ===>
     Member   ===>

EXEC Job Step ===>             Proc Step ===>

------------------------- Execution Overrides -------------------------------

PGM   ===> DFSRRC00            Initial Commands ===>

PARM  ===> BMP,PAYROLL,,,,N0003,,2,,,0002,,,IMST
 or Number of Linkage Parameters ===>

Task Libraries  ===> 'IMS.RESLIB'
   DSNAME       ===> 'USER01.LOAD'
     or         ===>
  (DDNAME)      ===>
                ===>
```

From the Monitor Control panel, the process to test a BMP is the same as the process to test an IMS batch program.

Define Your Application

At least one BMP must be added to the IMS/VS Stage 1.

Ask the appropriate technical staff member to add the following application definitions to your IMS/VS System:

1. The APPLCTN MACRO must be defined with the DOPT option so that a new copy of the PSB is obtained each time the program executes.
2. The PARMLIM must be zero. The option PGMTYPE=BATCH forces the PARMLIM to zero.
3. Define a different BMP for each applications programmer who is testing in this manner. Each programmer should be assigned a unique BMP to avoid having one programmer overwriting the PSB of another.

Note: You should only use this on an IMS/VS test system. The DOPT option can have a performance impact on a production IMS/VS system.

Test Your Application

The BMP defined is used to test the online application. Perform the following steps:

1. Replace the PSB for the BMP in the dynamic PSBLIB with the application PSB using the BMP PSB name.

   For example, the BMP PSB is ABCD1234 and the application PSB is MYAPPLTN. Thus, the PSB defining MYAPPLTN should be linked into the dynamic PSB library with a name of ABCD1234.

   The programmer can use the same BMP name for any IMS/VS online applications to be tested and debugged.

   Note: The dynamic PSBLIB data set must be concatenated after the ACBLIB data set. The PSB must reside in a library with the same format as IMSVS.ACBLIB and be concatenated to the IMSACB DD statement. The dynamic PSB must not be a member of IMSVS.ACBLIB.

2. Log on to the IMS/VS test system.

3. Enter an IMS/VS SET command in the form:

   ```
   /SET TRAN trancode
   ```

   where *trancode* is the trancode of the BMP defined in the previous section. This command directs all input messages from this LTERM (logical terminal) to *trancode*. Normally MFS provides the trancode to IMS/VS. The trancode of the application to be tested is different from the *trancode* through which the application Batch gains control.

4. Enter as many transactions as necessary. The transactions remain queued on the IMS/VS message queue until a test session is initiated.

5. Log off IMS/VS. You may have a VTAM product, which lets you do a VTAM switch without logging off IMS/VS.

6. Log on to TSO.

7. Begin a session with the following entries on the Execution Control panel:

   a. The name of the first module to be executed is DFSRRC00

   b. The execution parameters for the first program are as follows:

      BMP,mbr,psb,in,,ostd,,stimer,,,cputime,,,imsid

      where:

      - mbr-The name of the application program to be tested
      - psb-Psbname of the BMP defined previously containing the PSB of the application program being tested
      - in-The trancode of the BMP defined previously

- ostd-&opt&spie&test&dirca
  - &opt&spie&test&dirca
  - &opt - Action to be taken if the batch message region starts and no control program is active. Default is N (ask operator for decision).
  - &spie - Turn on the SPIE option (0) or not (1)
  - &test - Check (1) or not (0) the addresses in the user's call list for validity
  - &dirca - Required for dynamic PSBs
- stimer-Whether to set the timer (1) or not (0). Must be set to 1 if CPUTIME 0
- cputime-BMP task timing option. No timing (0) or maximum task time (*n*) where *n* is from 1 to 1440 minutes
- imsid-The IMS/VS system identifier where you want the BMP to executeC

8. End the session, after testing and debugging with the application.
9. Log off TSO.
10. Log on to IMS/VS to the LTERM used in Step 3.
11. Review the output messages that may have been sent to your LTERM from your application program.
12. Enter the IMS/VS RESET command as follows:

    ```
    /RESET
    ```

Application Testing Scenario

If you have two programmers testing online IMS/VS applications, you could define the following MACROS:

```
APPLCTN DOPT,PSB=INT01,PGMTYPE=BATCH
TRANSACT CODE=(INT01A),PRTY=(0,0)
APPLCTN DOPT,PSB=INT02,PGMTYPE=BATCH
TRANSACT CODE=(INT02A),PRTY=(0,0)

(PRTY=(0,0) is the default when PGMTYPE=BATCH.)
```

Jones would like to test and debug program PAYROLL while Smith tests program MANUFACT. Jones would either do a PSBGEN or copy his PSB (PAYROLL) into the dynamic PSBLIB member INT01. Likewise, Smith would do the same thing for his PSB (MANUFACT) but as member INT02.

Jones logs on to IMS/VS. He enters:

```
/SET TRAN INT01A
```

He then enters his FORMAT and data to execute his transaction. He can enter as many transactions as he likes. When he is done, he may either log off of IMS/VS or do a VTAM switch. He then logs on to TSO to start his test session. On the Execution Control panel under EXECUTION PARAMETERS he enters the following:

```
PARM ==> BMP,PAYROLL,INT01,INT01A,,N00003,,2,,,0002,,,IMST
```

When he has ended the application, he can log back on to IMS/VS and view any output messages that his application generated to his LTERM. Once he has viewed all of the IMS/VS output screens, he enters the following command before logging off IMS/VS:

```
/RESET.
```

Smith logs on to IMS/VS. He enters the following command:

```
/SET TRAN INT02A
```

He then enters his FORMAT and data to execute his transaction. He can enter as many transactions as he likes. When he is done, he can either log off IMS/VS or do a VTAM switch. He then logs on to TSO to start his test session. On the Execution Control panel under EXECUTION PARAMETERS he enters the following:

```
PARM==> BMP,MANUFACT,INT02,INT02,,N00003,,2,,,0002,,,IMST
```

When he has ended the application, he can log back on to IMS/VS and view any output messages that his application sent to his LTERM. Once he has viewed all of the IMS/VS output screens, he enters the following command before logging off IMS/VS:

```
/RESET
```

Since IMS/VS sees the program with a trancode other than the trancode of the program being tested, different users may schedule the program into multiple IMS/VS regions.

Test Message Switching Applications

Testing an IMS/VS application whose input messages are placed on the message queue by another IMS/VS program requires a modification to the procedure outlined in the section Testing Your Application. Remember that you use the /SET TRAN trancode command on your IMS/VS terminal to direct IMS to queue all transactions entered from your LTERM to the trancode specified in the SET command. Therefore, your application running as BMP transaction *trancode* is able to retrieve any messages on the queue destined for your application.

Note: The suggested method for testing IMS applications is using Option 5, Batch Link.

Unfortunately, any messages inserted to the message queue destined for another program will not be tagged with the BMP trancode. There is a way to work around this situation, as follows:

1. Follow steps 1 through 7 in the section Testing Your Application.
2. Set a breakpoint before the call to insert the message on the message queue. This is usually after the CHNG call.
3. Display the contents of the parms passed to your program using the DI or LI command.
4. Modify the trancode on the message to be inserted to be the BMP trancode and return to the Intercept panel.
5. Continue the testing of the application.
6. Return to the Execution Control panel, split the screen, and select Option 6 (TSO COMMAND PROCESSOR).
7. Copy the PSB for the program that retrieves the inserted messages into the dynamic PSBLIB data set with the BMP PSB member name. End the split screen mode.
8. Change the second execution parameter, mbr, to the name of the program retrieving the inserted messages, on the Execution Control panel.
9. Begin testing the application. Follow steps 8 through 12 in the section Testing Your Application.

Test Procedure Using BTS

Like an IMS batch application, a DB/DC application executed using BTS is not the first program executed. The module BTSTSOST is the first module executed. It in turn attaches the IMS region controller, DFSRRC00, which attaches the application program to be executed. To facilitate testing of IMS applications with both CA InterTest Batch and BTS, create a CLIST to allocate IMS, BTS, and CA InterTest Batch-related data sets. If you need help in creating the CLIST, ask your technical support staff for assistance.

The CLIST allocations are given as follows:

IMS-related:

```
DFSRESLB  IMS  IEFRDER  DFSVSAMP
```

Note: On the ddname DFSRESLB, the IMS RESLIB data set is concatenated to itself. This is necessary to prevent a dynamic allocation to the ddname. If the IMS RESLIB data set is not concatenated to itself under the ddname DFSRESLB, a USER 0684 abend, and other unpredictable user abends can occur.

BTS-related:

```
BTSIN BTSOUT BTSPUNCH BTSSNAP QIOPCB QALTPCB QALTRAN TASKLIB
```

CA InterTest Batch-related:

```
INT1PARM  INT1LOAD  INT1PNNL  INT1PROF  INT1MSGL  INT1CLOG INT1CLIB INT1REPT
```

Note: For additional information on CA InterTest Batch-related CLISTs, see the appendix "Basic Foreground Demo Session."

Preparation for Test Execution - BTS

Replace **Step 5** in the section Preparation for Test Execution with the following step:

- On the Execution Control panel, specify the name of the load library containing the BTS load modules. The first program to be executed is BTSTSOST.

  The execution parameters are the same parameters that are specified to BTS. Generally, the parameter specified is **DLI,,,**. If you specify the program member name, the psbname, or both, as you would to test a batch IMS program, a USER 0642 abend can occur (parm exceeds maximum length). The programs and the PSBnames to be tested are identified on ./T cards in the BTSIN data set.

Note: The dynamic STEPLIB function is performed by the BTS Tasklib option. Do not specify a TASKLIB ddname on the Execution Control panel, or BTS will believe that TASKLIB ddname is not allocated and terminate. Allocate the ddname TASKLIB in the CLIST to invoke the application. The load libraries containing the modules to be tested must be allocated to the ddname TASKLIB.

Use the Monitor Control panel to identify the programs that have their execution controlled by the application. Identify only the programs that you wish to debug. Also, provide the PROTSYM files that contain the symbolic information for the programs that you would like to debug.

BTS Interactions

After you have entered the Execution Control panel and the Monitor Control panel, BTS begins execution by reading the BTSIN data set. A BTS0007I message and images of the BTS input commands are displayed on the terminal. BTS will request a BTS command, a /FORMAT statement, or a /*. The /FOR modname causes BTS to display a formatted IMS screen for input. Once the data has been input, BTS attaches the application program. CA InterTest Batch gains control. When the application program calls CBLTDLI, BTS again has control until the IMS language interface module passes control back to the calling program. As a result, the BTS displays are interleaved with the Intercept panels.

If the BTS session ends prematurely and there was no BTS error message displayed on the terminal, review the BTSOUT data set. The BTSOUT data set will often contain data that does not appear on the terminal.

Note: The application intercepts all keyed PA1s during the BTS testing session.

Message Switching

Message switching is when an IMS application program inserts a message to the message queue that is either an output message to another IMS terminal or an input message to another transaction. If the message is an input message to another transaction, then BTS schedules another program to process the input message. To set breakpoints in the IMS program that processes the message inserted into the message queue, perform the following steps:

1. Program A inserts a message for program B. When the first executable statement in program A displays on the terminal, enter **QUALIFY B** on the command line. The beginning of the listing for program B displays. Set any UNCOND or WHEN breakpoints that you desire. Enter **QUALIFY A** on the command line to return to the program A display.
2. You can also accomplish this by specifying UNCOND ALL ENTRY on the command line of the Intercept panel upon initial entry into the first IMS application program.

BTS0067I MESSAGE

If this message should occur while testing in a BTS environment, make sure the BTSEXT parameter has been added to the PARMLIB member used to invoke the application. For more information, see Screen Handling Solution for the Message BTS0067I in the chapter "Installation" of the *Installation Guide*.

Display and Figure Verbs in a BTS Environment

When using these verbs in a BTS message switching environment, make sure the BTSSYSO parameter has been added to the PARMLIB member used to invoke the application. For more information, see Applications Using Message Switching and DISPLAY or Figure Verbs in the *Installation Guide*. Without this option, abends 0C4 and/or 0C1 can occur.

Test COBOL Programs Which Run Under an ADF Shell

Use this product to test COBOL programs that run under an ADF *shell*. The recommended procedure for doing this is the same as for testing COBOL programs under BTS. A programmer specifies BTSTSOST on the Execution Control panel as shown in the previous section and the names of the COBOL programs to be tested on the Monitor Control panel. Both of these panels are shown in the previous section Preparation for Test Execution - BTS. Specify the four-character ID of the ADF SIGN-ON screen on the /FOR statement. All of the functions of the application are available while testing the specified COBOL programs.

Although this application does not display or intercept code written in the ADF Audit Language, some users use the application facilities to help debug their ADF code. One way is to insert a call to a dummy COBOL program at whatever point in the ADF code they want to intercept execution. Once in the Breakpoint Intercept panel, you can use the Option 2: CORE to display ADF work areas and tables. This requires a good knowledge of ADF. The best source for this information is the IBM IMS ADF II Manuals.

Test IBM DB2 Applications

To test IBM DB2 applications using your product in foreground, you can modify the TSO CLIST MR91CDB2, which invokes the application. You find MR91CDB2 in the CAMRCLS0 data set. A sample *batch* execution of DB2 programs follows to give you some background into the DB2 commands DSN and RUN and how they relate to DB2 foreground execution.

Note: The instructions in this chapter are for debugging an application under Foreground, Option 1 on the Primary Option Menu. The suggested method for testing DB2 applications is using Option 5, Batch Link. For more information on batch link and about debugging your DB2 Stored Procedure, see the chapter "Batch Link Facility."

Note: If you are using the Call Attach Facility of DB2, you may use the MR91CLST CLIST to debug your application in foreground.

To run the DB2 program DSN8BC3 under CA InterTest Batch

1. Modify the TSO CLIST MR91CDB2 by replacing INTBATCH with the data set prefix defined at install time.
2. Verify that the application's ISPF Panel library is concatenated to the IBM ISPF panel library, ddname *ISPPLIB*, and the message library is concatenated to the IBM ISPF message library, ddname ISPMLIB.

Note: For more information, see MR91CDB2 in the section Customize CLISTs of the chapter "Installation" of the *Installation Guide.*

To test a DB2 program (PAYROLL in this example), perform the following steps:

1. EXEC the modified application CLIST.

   The DB2 panel displays.
2. Complete the three fields on the CA InterTest Batch DB2 panel. Each field is defined in the following table:

Field	Description
DB2 Subsystem Name ===>	The name of the DB2 Subsystem where the program being tested runs.

Field	Description
Plan Name ===>	The DB2 Plan name for the program being tested.
Library Name ===>	The Load Library name where the program being tested resides.

3. Press Enter after completing the panel fields. The Primary Option Menu displays.
4. Select Option 1: Foreground. The Execution Control panel displays.
5. Supply the main application program name and the required STEPLIB to execute the application. Press Enter to see the Monitor Control panel.
6. On the Monitor Control panel, type in the names of the programs that you want to debug and DSNs of the PROTSYM files containing the symbolic information for the programs you want to debug. Press Enter to receive the *INITIAL* Intercept panel.
7. Run the test session in the usual way.

   Notice that once you invoke CA InterTest Batch, there is no difference in the way the application is used to test DB2 programs.

   Note: This procedure does not let you end the program and return to the Execution Control panel to retest the same program. You must exit the application and rerun the CLIST (Step 1).

Use the following approach when debugging in foreground with DB2 and CA InterTest Batch:

1. Get CA InterTest Batch working for non-DB2 programs.
2. Get the DB2 program running under native TSO.
3. Ensure that the DB2 program is correctly prepared as follows:
   a. DB2 pre-processed
   b. Compiled and post processed into a PROTSYM file
   c. Linked
   d. A DB2 bind step or job is run for the recompiled program
   e. All other DB2 protocol is done
4. Modify the application CLIST as explained previously.

Test Data Generation

This product assists you in generating test data or files for testing. Consider the following COBOL program:

```
000100 IDENTIFICATION DIVISION.
000200
000300 PROGRAM-ID.    DATAGEN.
000400 AUTHOR.        J. DOE.
000500 INSTALLATION.  DATACEN.
000600 REMARKS.
000700
000800
000900 ENVIRONMENT DIVISION.
000901 INPUT-OUTPUT SECTION.
000902 FILE-CONTROL.
000910 COPY OUTFILE.
001000
001100 DATA DIVISION.
001200
001300 FILE SECTION.
001400 COPY OUTREC.
001500
001600 PROCEDURE DIVISION.
001700 BEGIN.
001800     OPEN OUTPUT OUT-FILE.
001900 0100-WRITE-IT.
002000     WRITE OUT-RECORD.
002010     GO TO 0100-WRITE-IT.
002100 0100-END-IT.
002200     CLOSE OUT-FILE.
002300     GOBACK.GOBACK Key
```

Assume that the COPY codes OUTFILE and OUTREC contain the SELECT and FD information respectively. Use the following command to set a breakpoint at the write statement and display the record contents in COBOL format:

```
UNCOND 0100-WRITE-IT (RECORDS)
```

Each time statement 1900 is executed, the COBOL program stops and a formatted display of the record for OUT-FILE appears. To generate the data, type over the values that you want, press the END key to return to the Intercept panel, and press the GO key to write the record.

Adding another File Definition, OPEN, and a READ before the WRITE could make this into a COBOL format record editor.

How to Test Under CA Roscoe

If you run this product under CA Roscoe, you should be aware of the following points:

- The suggested method for testing applications with CA Roscoe is using Option 5, Batch Link.
- The application runs under the ETSO OR BTSO facility of CA Roscoe.
- A special JCL conversion facility to simplify file allocations is available for CA Roscoe users and is described in the chapter "JCL Conversion for CA Roscoe."
- Your application PF key assignments can differ from your CA Roscoe PF key assignments. Define PF keys by selecting the KEYS option on the Primary Option Menu.
- The application lets you use the split-screen facility to establish a second session using the PF SPLIT key. You also can use standard CA Roscoe commands to initiate a second CA Roscoe session.
- Use the CA Roscoe SUSPEND function to suspend a CA InterTest Batch session and return to CA Roscoe. You can then perform other CA Roscoe functions; however, you cannot call another ETSO OR BTSO application. Resume ETSO OR BTSO to return to your session.
- Use the PF PRINT key to print to a CA InterTest Batch data set, which can be allocated to the AWS or any MVS data set.
- Use the CA Roscoe print facility to print to a CA Roscoe data set.

Chapter 6: Allocations Facility for ISPF

Before you can begin a debugging session, you must allocate the data sets that your program will be using. The allocations facility for TSO/ISPF users provides a number of features that makes this set-up process easier.

Briefly, the allocations facility lets you to do the following actions:

- Create and manage ALIB (Allocations Library) allocations members. ALIB is the internal allocations format for the application. The ALIB is a PDS that consists of members created through the ALIB editor or JCL converter of this application. You can create new allocations library members, or edit, browse, allocate and free data sets specified in existing members. Once you save an ALIB, you can use its allocations in the future, saving you valuable debugging set-up time.

  Note: The ALIB dataset is optional and is used by the CA InterTest Batch foreground testing facility. For more information, see the section Allocate ALIBs (Optional) in the *Installation Guide*.

- Convert allocation lists from ALIB to CLIST format. Although ALIB is an efficient format that is sufficient for most situations, conversion to CLIST is useful in complex allocation situations that are beyond the straightforward free-and-allocate capabilities of ALIB (conditional data set allocations, for example). Converting to CLIST may also be necessary if your particular operating environment requires the CLIST format.

- Browse existing JCL members in PDS, CA Librarian, and CA Panvalet libraries.

- Convert JCL to ALIB format. This feature is useful if you have existing JCL that you would like to use for your allocations. The system's conversion utility fully supports JCL allocation features, and converted JCL is displayed in Edit mode for further processing. You can also use the resulting ALIB on the Execution Control panel to allocate and free the required application DDs. You can convert JCL to CLIST format by first converting it to ALIB format, then converting it from ALIB to CLIST (it is not necessary to save in ALIB format). Using this feature saves you the time needed to key in and debug ALIB or CLIST statements from scratch.

The allocations facility panels conform to ISPF file name and command conventions (for example, the ISPF Locate command).

Note: For more information on how to use the JCL conversion facility, CA Roscoe users should see the chapter "JCL Conversion for CA Roscoe."

This section contains the following topics:

The Allocations Main Menu

Access the Allocations Main Menu by selecting option 3 from the Primary Option Menu. The Allocations Main Menu the starting point for accessing all of the allocations facility's features:

The fields on the Allocations Main Menu are as follows:

1 ALIB	Create new allocations members, edit and browse existing members, allocate/free data sets specified in members, and convert ALIB members into CLIST format.
2 JCL	Browse existing JCL members in PDS, CA Librarian, or CA Panvalet libraries. You can also select members for conversion from JCL to ALIB format. The converted JCL is displayed in Edit mode for further processing.
3 CURRENT	Display current session allocations and free, browse, or edit allocated data sets.
4 OPTIONS	Specify various JCL conversion options and ALIB member export options, and identify procedure libraries (proclibs) used for expanding JCL procedures found during JCL conversion.

Option 1: ALIB (Allocations Library) Functions

The ALIB library option lets you create new allocations members, edit and browse existing members, allocate/free data sets specified in members, export (convert) ALIB members to CLIST format, and import (convert) JCL members to ALIB format. Selecting this option displays the Allocations Library panel.

Select an ALIB Member from the Allocations Library Panel

The application displays the Allocations Library panel if you select option 1 (ALIB) from the Allocations Main Menu. Use this panel to enter the name of the member you want to edit.

For ISPF library members: The Allocation Library panel has the following input fields:

For ISPF library members:

Project, Group, Type Required. Enter the three-level qualifier for standard data set names in ISPF.

Member Optional. If you do not know the member name, leave this field blank or enter a member name pattern. Valid ISPF *wild card* member pattern characters are:

- **%** Substitutes for any one character. May be the first character.
- ***** Substitutes for any number of characters. May not be the first character.

For PDS members:

Data set name Optional. If your member is not in an ISPF library, enter the actual, cataloged name of the file. The name can be up to 44 characters long, with segments up to eight characters long, separated by periods. You can specify any MVS data set name and use wild card substitutions, as described for the Member field.

Volume serial Optional. This field identifies the VOLSER of the data set specified in the data set name field and is necessary only when a file is not cataloged.

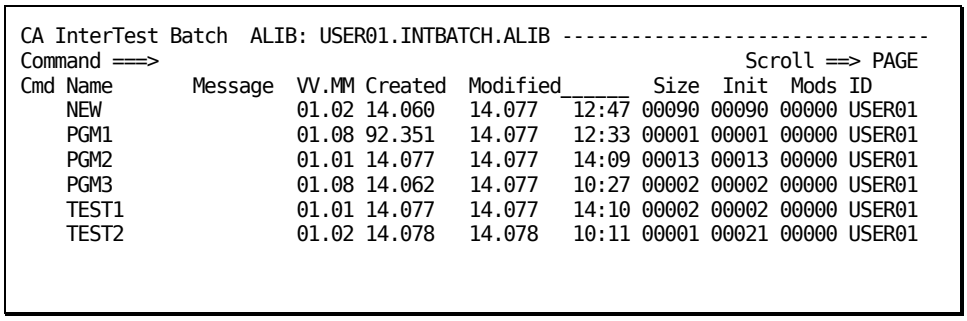
To select an ALIB Member from the Allocations Library panel:

1. Enter the ISPF library or PDS information for the member you want to edit.
2. Press Enter.

If you entered a complete member name, the application displays the Allocations Edit panel. If you entered a partial member name, the application displays the ALIB Member Selection List.

Select an ALIB Member from the ALIB Member Selection List

If you did not specify a complete member name on the Allocations Library panel, the application displays the ALIB Member Selection List panel. This panel lists all the allocations library members or those that match the pattern.



```
CA InterTest Batch  ALIB: USER01.INTBATCH.ALIB --------------------------------
Command ===>                                                  Scroll ==> PAGE
Cmd Name       Message  VV.MM Created  Modified_____  Size  Init  Mods ID
    NEW                 01.02 14.060   14.077   12:47 00090 00090 00000 USER01
    PGM1                01.08 92.351   14.077   12:33 00001 00001 00000 USER01
    PGM2                01.01 14.077   14.077   14:09 00013 00013 00000 USER01
    PGM3                01.08 14.062   14.077   10:27 00002 00002 00000 USER01
    TEST1               01.01 14.077   14.077   14:10 00002 00002 00000 USER01
    TEST2               01.02 14.078   14.078   10:11 00001 00021 00000 USER01
```

Enter the following line commands in the Cmd field before each member in the list:

AL To allocate all the data sets in the ALIB member.

B To browse the ALIB members. The Allocation List Browse panel displays. From this panel you may view the allocation list in an ALIB. You can scroll through the members and use the ISPF Locate (L) command to locate specific members. The Allocation List Browse panel is identical in format to the Allocations Edit panel, but no edits may be made.

C To display the Convert to CLIST panel, which lets you convert the selected ALIB member to CLIST format. For more information, see Converting ALIB to CLIST (Exporting ALIB).

E/S To edit the ALIB member. Issuing this command displays the Allocations Edit panel. For more information, see Editing ALIB Allocations.

F To free all the ddnames in the ALIB member.

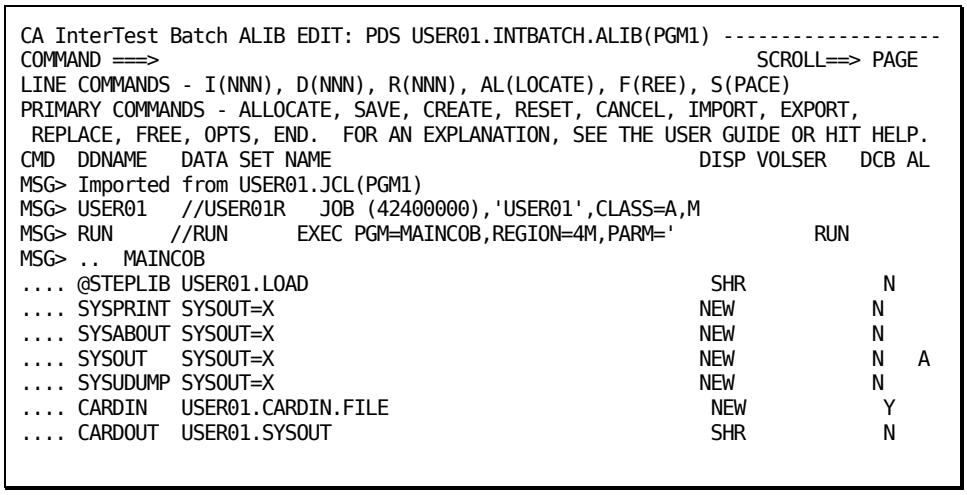
Edit ALIB Allocations

Display the Allocations Edit panel in any one of the following ways:

- From the Allocations Library panel by specifying a particular member
- From the ALIB Member Selection List panel by entering the line commands E or S next to a member
- From the JCL Allocations Step Selection List panel by selecting converted JCL steps for allocation

The Allocations Edit panel lets you create, modify, and manipulate lists of data set allocations. A list may be saved into the ALIB, exported (converted) into a CLIST library, allocated, or freed. JCL may be imported (converted) into ALIB format.

A sample Allocations Edit Panel is shown next:



```
CA InterTest Batch ALIB EDIT: PDS USER01.INTBATCH.ALIB(PGM1) -------------------
COMMAND ===>                                                  SCROLL==> PAGE
LINE COMMANDS - I(NNN), D(NNN), R(NNN), AL(LOCATE), F(REE), S(PACE)
PRIMARY COMMANDS - ALLOCATE, SAVE, CREATE, RESET, CANCEL, IMPORT, EXPORT,
 REPLACE, FREE, OPTS, END.  FOR AN EXPLANATION, SEE THE USER GUIDE OR HIT HELP.
CMD  DDNAME   DATA SET NAME                                DISP VOLSER   DCB AL
MSG> Imported from USER01.JCL(PGM1)
MSG> USER01   //USER01R   JOB (42400000),'USER01',CLASS=A,M
MSG> RUN     //RUN      EXEC PGM=MAINCOB,REGION=4M,PARM='            RUN
MSG> ..  MAINCOB
.... @STEPLIB USER01.LOAD                                   SHR            N
.... SYSPRINT SYSOUT=X                                     NEW            N
.... SYSABOUT SYSOUT=X                                     NEW            N
.... SYSOUT   SYSOUT=X                                     NEW            N   A
.... SYSUDUMP SYSOUT=X                                     NEW            N
.... CARDIN   USER01.CARDIN.FILE                            NEW            Y
.... CARDOUT  USER01.SYSOUT                                 SHR            N
```

Allocations Edit Panel fields are described next:

CMD Enter all line commands in this area.

DDName Data definition name. Enter a valid ddname in this field.

Data Set Name Data set name. Enter a valid data set name in this field. Data set names must be fully qualified and without apostrophes. SYSOUT= and TERMINAL are valid entries.

Disp The disposition of the data set. Enter the disposition of the data set in this field. Valid dispositions are:

OLD
SHR
NEW

Default: SHR.

Volser The old volume serial, if any, of the data set. You may enter another volume serial on which the data set is to reside.

DCB

Yes	DCB information is present
No	There is no DCB information for the data set

AL

A	The ddname is allocated
blank	The ddname is not allocated

Primary Commands

The Allocations Edit Panel primary commands, which you enter on the COMMAND ===> line, are shown next.

ALLOCate Allocates all the allocations listed on the screen.

SAVE Saves any changes made to the list of allocations in the current ALIB member.

CREate Writes the list of allocations to a new ALIB member. Issuing this command displays the Copy to Allocation Library panel.

RESet Removes all non-relevant MSG lines created during JCL conversion or during allocation.

CANcel Exits without saving any changes and redisplays the Member Selection List panel from which you entered the editor. All changes to the member that were not saved are discarded.

IMPort Invokes the JCL Converter to convert raw JCL into ALIB format. Import JCL from a PDS, CA Librarian, CA Panvalet, and current session allocations libraries. The application displays the JCL Import Specifications panel. For more information, see Converting JCL to ALIB (Importing JCL).

EXPort Converts the current ALIB member into CLIST format. For more information, see Converting ALIB to CLIST (Exporting ALIB).

REPlace Replaces a member with the contents of the current member. Issuing this command displays the Copy to Allocation Library panel.

END Ends the current Edit session and displays the Member Selection List panel from which you entered the editor. The member is automatically saved if it was modified.

FREE Frees all ddnames in the current member.

OPTS Lets you change conversion and allocation options. Invoking this command displays the JCL Conversion/Allocation Options panel. For more information, see Option 4: JCL Conversion/Allocation Options.

Line Commands

Enter line commands in the Cmd field next to the ddname:

Innn Inserts one or more blank edit lines. *nnn* is the number of lines to insert. The default is 1 and the maximum is 999. Any blank edit line with no data on it is deleted the next time you press Enter.

Example: I3 Inserts 3 blank lines.

Dnnn Deletes one or more edit lines. *nnn* is the number of lines to delete. The default is 1 and the maximum is 999.

Example: D Deletes this line.

Rnnn Repeats (duplicates) the line *nnn* times. The default is 1 and the maximum is 999.

Example: R99 Duplicates this line 99 times.

AL Allocates the data set. If the allocation succeeds, an **A** appears in the AL field. If the allocation fails, one or more messages will be inserted after the line.

Example: AL Allocates this data set.

C Changes the current line to a comment.

Note: Once a DD is converted to a comment, it can no longer be edited.

F Frees (deallocates) the data set specified in the line. Upon deallocation, the **A** in the AL field disappears.

Example: F Frees this data set.

S Displays the Data Set Attributes panel, which contains DCB and space information, and extended (SMS) attributes.

Dataset Attributes Panel

Use the Dataset Attributes panel to review and change the data set attributes including DCB information, space information, and extended (SMS) attributes. This panel displays when you select a ddname on the Allocations Edit panel.

Dataset Attributes Panel: Fields

The following table describes the Dataset Attributes Panel fields:

LRECL The logical record length of the data set.

BLKSIZE The block size of the data set.

RECFM The record format of the data set. Valid formats are:

- **F** Fixed
- **FB** Fixed blocked
- **FBA** Fixed blocked with carriage control character
- **V** Variable
- **VB** Variable blocked
- **U** Undefined

Default: the current record format of the data set.

DSORG The organization of the data set. Valid organizations are:

- **PO** Partitioned Organization (PDS)
- **PS** Sequential

SPACE UNIT The space unit used by the data set for primary and secondary storage. Valid space units are:

- **BLK** Blocks
- **TRK** Tracks
- **CYL** Cylinders

Default: the current space unit of the data set.

PRIME SIZE The primary storage.

SECONDARY The secondary storage.

DIRECTORY BLOCKS The number of directory blocks.

RELEASE Whether unused storage should be released. Valid values are:

- **Yes** Release unused storage
- **No** Do not release unused storage

Default: No

UNIT NAME The unit name.

VOLUME SERIAL If you entered a VOLSER on the Allocations Edit panel, it is displayed here. If this field is blank or you want to change the VOLSER, enter the new VOLSER.

OR ALLOCATE LIKE DDNAME OR DSNAME Use this field to obtain DCB information from another data set or ddname. Enter a valid ddname in parentheses (for example,INT1LOAD), or a fully qualified data set name.

STORCLAS The SMS storage class.

DATACLAS The SMS data class.

MGMTCLAS The SMS management class.

DSNTYPE One of the following valid data set types:

LIBRARY
PDF
PIPE
HFS

RECORG One of the following valid record organizations:

- **KS** Keyed sequential file
- **ES** Entry-sequenced file
- **RR** Relative record file

	LS	Linear space file
AVGREC	Allocate space by records using one of these valid specifications:	
	U	Unit is one record
	T	Unit is one thousand records
	M	Unit is one million records
KEYOFF	Identifies the offset within the record where the key resides, for keyed files only.	
KEYLEN	Identifies the length of the key.	

To change DCB information

1. Enter the DCB information in the appropriate fields.
2. Enter **END** on the command line and press Enter, or press the End key.

Result: The application saves the DCB information and redisplays the Allocations Edit panel.

Note: Enter **CANCEL** to ignore the DCB information and to redisplay the Allocations Edit panel.

Copy to Allocation Library Panel

The application displays the Copy to Allocation Library panel whenever the ALIB editor needs the name of the ALIB data set and member in order to write the allocations list. The application displays this panel if you issue a Replace or Create command from the Allocations Edit panel.

To specify the ALIB data set name

Note: The ALIB data set is optional. If your installation is not using an ALIB data set no entry is required on this panel. Press PF3 several times until you return to the CA InterTest Batch PRIMARY OPTION MENU. For more information, see the section Allocate ALIBs (Optional) in the *Installation Guide*.

1. If the data set to which you are writing is an ISPF library, complete the information for the ISPF Library fields (Project, Group, Type, and Member). For more information on these fields, see Selecting an ALIB Member from the Allocations Library Panel.
2. If the data set to which you are writing is not an ISPF library, complete the information for the Other partitioned data set fields (Data set name, Volume serial). For more information on these fields, see Selecting an ALIB Member from the Allocations Library Panel.

3. Enter **Yes** in the Replace like-named member field if you want the editor to overwrite members by the same name in the library (the default is No).
4. Press Enter.

Result: The application creates or replaces the ALIB data set and redisplays the Allocations Edit panel.

Convert ALIB to CLIST (Export ALIB)

Access the Convert to CLIST Options panel in one of these ways:

- From the ALIB Member Selection List panel by entering the line command C (Convert to CLIST) next to the member to convert.
- From the Allocations Edit panel by entering the primary command EXP (Export) on the command line.

If your allocation situation is too complex for the limitations of the ALIB format (for instance, if you need conditional allocations or need to use other TSO commands beyond simple free-and-allocate statements), you may find it useful to translate your ALIB statements to CLIST.

The ISPF library and Other partitioned data set fields, in which you specify your destination (*TO*) data set, are the same as those on the Allocations Library panel.

The other fields are as follows:

Replace ? Determines if the target CLIST library is to be replaced.

- **Yes** Replace the target CLIST library
- **No** Do not overwrite the existing CLIST library. After you press Enter, the cursor is placed in the Data Set Name field and you can specify a new name

Edit CLIST ? Determines if the ISPF editor will be invoked to edit the converted member.

- **Yes** The ISPF editor will be invoked to edit the converted member
- **No** The ISPF editor will not be invoked

To convert from ALIB to CLIST

1. Specify the destination (TO) data set in either the ISPF library or the Other partitioned data set fields. Modify the member name, if necessary.
2. Enter **Yes** or **No** in the Replace? and Edit CLIST? fields.
3. Press Enter.

Result: The application performs the conversion. If you entered Yes in the Edit CLIST field, the application invokes the ISPF editor to allow you to further modify the CLIST. If you entered No in the Edit CLIST field, the application redisplays the panel from which the converter was invoked.

Convert JCL to ALIB (Import JCL)

The application displays the JCL Import Specifications panel when you use the Import command on the Allocations Edit Panel. Use this panel to enter the name and type of the JCL library member from which you want to import JCL for conversion.

The ISPF library and Other data set fields are the same as those on the Allocations Library panel. The other fields are as follows.

Option The data set format of the data set you want to import for conversion.

- **Blank** The member is in a PDS.
- **L** The member is in a CA Librarian library.
- **P** The member is in a CA Panvalet library.
- **S** Import the currently allocated data set list (current session allocations). No other fields on this panel need to be completed if you choose this option.

CA Librarian index type When converting CA Librarian members, determines the amount of directory information displayed.

- **Q** Quick. Displays the member name only and takes less time than the S (Short) option.
- **S** Short. Displays more information on the selected member than the Q (Quick) option.

To convert from JCL to ALIB

1. Enter the data set format of the JCL member you want to import for conversion.
2. In either the ISPF library or the other partitioned data set fields, specify the source *(FROM)* data set that contains the JCL you want to import for conversion.
3. Press Enter.

Result: If you have requested the JCL Conversion/Allocations Options Panel, it will be displayed next. Otherwise, the JCL member selection panel is displayed.

Option 2: The JCL Library/Allocations Functions

Option 2 (JCL) on the Allocations Main Menu panel lets you select existing JCL members for conversion to ALIB format. PDS, CA Librarian, and CA Panvalet libraries are supported. Use the resulting ALIB member on the Execution Control Panel to allocate and free DDs.

Specify the JCL to Convert

The application displays the JCL Library panel when you select the JCL option from the Allocations Main Menu. Use this panel to specify the name and type of the JCL member you want to convert.

Complete the following input fields:

Option The format of the data set you want to import for conversion.

- **Blank** The member is in a PDS.
- **L** The member is in a CA Librarian library.
- **P** The member is in a CA Panvalet library.

For ISPF Library members:

Project, Group, Type Required. Enter the three-level qualifier for standard data set names in ISPF

Member Optional. If you do not know the member name, leave this field blank or enter a member name pattern. Valid ISPF *wild card* member pattern characters are:

- **%** Substitutes for any one character. May be the first character.

*	Substitutes for any number of characters. May not be the first character

For PDS members:

Data set name Optional. If your member is not in an ISPF library, enter the actual, cataloged name of the file. The name can be up to 44 characters long, with segments up to eight characters long, separated by periods. You can specify any MVS data set name. You can use wild card substitutions, as described for the Member field.

Volume serial Optional. This field identifies the VOLSER of the data set specified in the Data set name field and is necessary only when a file is not cataloged.

CA Librarian index type Used only when converting CA Librarian files. Determines the amount of directory information the application displays.

- **Q** Quick. Displays the member name only and takes less time than the S (Short) option.
- **S** Short. Displays more information on the selected member than the Q (Quick) option.

To specify a JCL member for conversion:

1. Enter the ISPF library or PDS information.
2. If you are converting a CA Librarian member, enter the CA Librarian index type you desire.
3. Press Enter.
4. The application displays the JCL Member Selection List panel.

Use the JCL Member Selection List

After specifying a partial or complete member name on the JCL Library panel, the application displays the JCL Member Selection List panel.

```
CA InterTest Batch JCL: PDS USER01.INTBATCH.JCL---------------------------------
Command ===>                                                 Scroll ==> PAGE
    Enter  B to Browse member, S to select member for JCL conversion
Cmd Name       Message  VV.MM Created  Modified_____  Size  Init  Mods ID
    ABSDUMP             01.00 14.325   14.325   15:47 00006 00006 00000 USER01
    ASMHAL              01.00 14.154   14.154   11:40 00015 00015 00000 USER01
    TESTX               01.02 14.008   14.211   15:47 00043 00022 00027 USER01
    TEST2               01.03 14.188   14.062   09:13 00005 00005 00000 USER01
    TEST3               01.00 14.188   14.188   10:34 00007 00007 00000 USER01
```

This panel displays the JCL members that match the member name or wild card pattern specified in the Member field on the JCL Library panel. If you made no entry in the Member field, the application displays all the members of the library.

Enter the following line commands in the Cmd field before each member in the list:

B To browse the JCL library member. the application displays the JCL Browse panel.

S To select the JCL member for conversion to ALIB format. Once the JCL conversion has ended, the application displays the JCL Allocations Step Selection List panel. Note that if you have not set the option to suppress the Conversion/Allocation Options panel, it is displayed before converting the JCL and displaying the JCL Allocations Step Selections Panel. For more information, see Option 4: JCL Conversion/Allocation Options.

The JCL Browse Panel

The application displays the JCL Browse panel when you use the Browse command on the JCL Member Selection List panel.

Use the JCL Browse panel to check the contents of a member selected for conversion and to determine the STEP names. Scroll through the member to find the STEPs you want to use. Entering an END command or pressing the End key redisplays the JCL Member Selection List panel.

```
CA InterTest Batch Browse JCL: PDS USER01.INTBATCH.JCL --------------------------
Command ===>                                                  Scroll ==> PAGE
Cmd _____________________________________ JCLs ___________________________________
    //USER01C  JOB (12300000,76),'RUN',CLASS=A,MSGCLASS=Z,
    //    NOTIFY=USER01
    //PROCLIB  DD  DISP=SHR,DSN=USER01.PROCLIB
    //QDD1     EXEC  COB,N=QDD
    //COB.SYSIN DD   DISP=SHR,DSN=USER01.QDD.COBOL
    //QEE2     EXEC  COB,N=UXX
    //COB.SYSIN DD   DISP=SHR,DSN=SYS2.ROSCOE56.DAILY.BKUP(-5)
```

Select Converted JCL Steps for Allocation

The application always displays the JCL Allocation Step Selection List panel after JCL conversion, if there were multiple steps in the JCL. The panel lists all JCL EXEC cards in the member:

```
CA InterTest Batch JCL: PDS USER01.INTBATCH.JCL(TEST3) -------------------------
Command ===>                                                  Scroll ==> PAGE
    Enter S beside step to select, or press END key to select all
Cmd---------------------------------- Steps -----------------------------
    DELETE    //DELETE  EXEC PGM=IEFBR14
    C         //C       EXEC PGM=IKFCBL00,REGION=4M,
    LKED      //LKED      EXEC PGM=IEWL,
    PRINTIT   //PRINTIT   EXEC PGM=IEBGENER,REGION=1024K
    DELETE    //DELETE  EXEC PGM=IEFBR14
    C         //C       EXEC PGM=IKFCBL00,REGION=4M,
    LKED      //LKED      EXEC PGM=IEWL,
    PRINTIT   //PRINTIT   EXEC PGM=IEBGENER,REGION=1024K
```

Note: If you have not set the option to suppress the Conversion/Allocation Options panel, it displays before converting the JCL and displaying the JCL Allocations Step Selections Panel. For more information, see Option4: JCL Conversion/Allocation Options.

To select converted JCL steps for allocation

1. On the JCL Allocation Step Selection List panel, enter **S** next to the steps whose data set allocations you want to use or enter the END command (or press the END key) to use all the steps. The steps you select form your allocation list.
2. Press Enter (unless you used an END key instead of entering the END command in step 1).
3. The application displays the Allocations Edit panel so you may edit the newly created Allocation List, saved in an allocation library or converted to CLIST format.

For more information on working with allocation libraries, see Editing ALIB Allocations.

The Conversion Process

When you select JCL for conversion to ALIB format, the JCL converter checks the JCL for syntax. If the JCL executes procedures and PROCLIBs were specified on the Conversion/Allocation Options panel, the JCL converter expands the procedures found in the PROCLIBs, substitutes parameters, and performs step overrides. If the JCL executes a procedure that is not found in a PROCLIB, a WARNING panel is displayed and JCLCheck message CAY6027E is included in the ALIB member, indicating the name of the procedure that was not found.

```
CA InterTest Batch ALIB EDIT: PDS CAI.CAMRJCL(DEMOJCL)                    ----
COMMAND ===>                                                 SCROLL==> PAGE
LINE COMMANDS - I(NNN), D(NNN), R(NNN), AL(LOCATE), F(REE), S(PACE)
PRIMARY COMMANDS - ALLOCATE, SAVE, CREATE, RESET, CANCEL, IMPORT, EXPORT,
 REPLACE, FREE, OPTS, END.  FOR AN EXPLANATION, SEE THE USER GUIDE OR HIT HELP.
CMD  DDNAME   DATA SET NAME                                 DISP VOLSER   DCB AL
MSG> Imported from CAI. CAMRJCL(DEMOJCL)
MSG> DEMOJCL //DEMOJCL  JOB (124400000),'INTBAT91 DEMOJCL
MSG> STEP1   //STEP1    EXEC PGM=CAMRCOBB,REGION=4M                STEP1
MSG> ..  .CA
.... @STEPLIB USER01.LOAD                                    SHR           N
.... SYSPRINT SYSOUT=X                                      NEW           N
.... SYSOUT   SYSOUT=X                                      NEW           N
.... REPORT   SYSOUT=X                                      NEW           N
******************************** BOTTOM OF DATA ********************************
```

To save the created Alib member

1. Place the cursor on the command line and type in 'SAVE' to save the ALIB version of the JCL.

   The Copy to Allocation Library screen opens.

2. Fill in the library and member name where your ALIB member is to be saved and press Enter.

   The ALIB member is saved.

Special Considerations

There are some special conversion considerations for certain types of DD statements. The application allocates DD statements to a particular TSO session by using SVC 99, which performs dynamic allocation. SVC 99 does not allow the following four DD statements to be dynamically allocated:

- STEPLIB
- JOBLIB
- STEPCAT
- JOBCAT

The application changes STEPLIB to @STEPLIB and JOBLIB to @JOBLIB.

JCL Keywords Supported for Conversion

The JCL converter supports the following JCL keywords and positional operands. The degree of converter support is noted.

JCL Keywords	Description
DSN=	Fully supported.

JCL Keywords	Description
DISP=	All operands except conditional dispositions are supported. If a DD with a conditional disposition is passed, the third sub parameter is ignored.
DCB=	The DCB keywords supported are: RECFM= BLKSIZE= LRECL= DSORG=
HOLD=	Ignored.
SYSOUT=	Fully supported. Only the CLASS parameter is used.
UNIT=	Fully supported.
VOL=	Fully supported. Only the VOLSER parameter is used.
SUBSYS=	Fully supported. Check with the vendor to make sure that a DD can be allocated for this subsystem with SVC 99.
SPACE=	Fully supported.
STORCLAS=	Fully supported.
DATACLAS=	Fully supported.
MGMTCLAS=	Fully supported.
DSNTYPE=	Fully supported.
RECORG=	Fully supported.
AVGREC=	Fully supported.
KEYOFF=	Fully supported.
KEYLEN=	Fully supported.
LIKE=	Fully supported.
REFDD=	Fully supported.

JCL Keywords Not Supported for Conversion

The JCL converter does not convert all JCL keywords because many parameters in batch JCL are not supported in TSO CLISTs. The converter ignores backward references and keywords that it does not convert.

The following DD statement keywords are not converted.

AMORG=	FLASH=
BURST=	FREE=

CHARS=
CHKPT=
CNTL=
COPIES=
DDNAME=
DEST=
DLM=
DSID=
FCB=
LABEL=
MODIFY=
MSVGP=
OUTLIM=
OUTPUT=
PROTECT=YES
QNAME=
UCS=

Option 3: Current Allocations

The application displays the Current Allocations panel when you select option 3 (CURRENT) from the Allocations Main Menu panel. The current allocations option displays current session allocations and lets you free, browse, or edit allocated data sets.

The Current Allocations panel lists currently allocated data sets, which may have been previously allocated in TSO Logon Procedures (such as STEPLIB), by a CLIST, or by the Allocation Facility. A sample Current Allocations List panel follows:

```
----------------- CA InterTest Batch Current Allocations List ------------------
Command ===>                                                  Scroll ==> PAGE
   Line commands: F - Free, B - Browse, E -Edit.
Cm DDName   Disp   Dataset Name                                          Volser
   ISP10155 SHR    USER01.INTBATCH.ISPMLIB                                OSI011
   ISPPROF  SHR    USER01.ISPF.ISPPROF                                    CAI801
   SYSPRINT        TERM
   SYSTERM         TERM
   SYSOUT          TERM
   SYSUDUMP        JES
   SYSIN           TERM
   SYS00001 SHR    SYS1.BRODCAST                                          MVSLIB
   ISPALIB  SHR    ISP.ISPALIB                                            MVXAD1
   ISRCFIL  SHR    ISR.ISPFLMF.CFIL                                       MVS003
   SYSHELP  SHR    SYS2.HELP                                              MVSLIB
            SHR    SYS1.HELP                                              MVXAD1
            SHR    USER.HELP                                              CAI800
   ISPTABL  SHR    USER01.ISPF.ISPPROF                                    CAI801
   ICQTABL  SHR    ICQ.ICQTLIB                                            MVXAD1
   SYSPROC  SHR    SYS2.CLIST                                             MVSLIB
            SHR    ICQ.ICQCCLIB                                           MVXAD1
            SHR    ICQ.ICQACLIB                                           MVXAD1
            SHR    ISR.ISRCLIB                                            MVXAD1
            SHR    SYS1.RMFCLS                                            MVXAD1
```

You may scroll through the list and enter the following line commands next to any data set in the list:

F/S Free the data set/ddname.

B Browse the data set. This is an ISPF function and operates only if it is supported by your ISPF installation.

E Edit the data set. This is an ISPF function, and operates only if it is supported by your ISPF installation.

Option 4: JCL Conversion/Allocation Options

The application displays the JCL Conversion/Allocation Options panel when you select option 4 (OPTIONS) from the Allocations Main Menu panel.

Use the Options option to specify JCL conversion and ALIB member export options and Proclibs (procedure libraries) used for expanding JCL procedures found during JCL conversion.

The Current Allocations/Options fields are described next:

Proclib 1-7 The procedure library names you want to use for expanding procedures found in JCL during conversion. When your JCL job stream executes procedures, assign the procedures a Proclib to be used during the conversion process. Enter up to seven procedure library names. If you require more than seven Proclibs, use a pre-allocated ddname instead.

Pre-allocated DD If you require more than seven Proclibs, or if your Proclibs are pre-allocated by other means (for example, in the TSO logon procedures or custom CLISTs), enter the name of a pre-allocated ddname here.

EasyProclib For EasyProclib users, this field specifies whether the JCL conversion facility uses private JCL procedure libraries identified by the //PROCLIB ddname in the job stream.

Yes	The JCL conversion facility uses private JCL procedures using the //PROCLIB ddname in the job stream.
No	The JCL conversion facility will not use private JCL procedures using the //PROCLIB ddname in the job stream.
Default:	No

CLIST 'Control' Whatever you enter here appears in the CLIST control statement when you convert from ALIB to CLIST (for example, NOMSG and NOLIST). For more information on valid CLIST control statements, see the IBM publication *TSO Command Language Reference Manual*.

DCB info on (Attr, Alloc)

Determines whether DCB information appears on the attribute statement or the allocate statement in your CLIST when you convert from ALIB to CLIST. The choice is mainly dictated by user-preference.

ALLOC	Put DCB information on the generated ALLOC statement.
ATTR	Generate DCB information on a separate ATTRIB statement.
Default:	Attr

If allocated (Reuse, Free)

Controls the allocation process during CLIST conversion. If the data set is already allocated:

R	The application tries to reuse the allocation. The REUSE statement is generated on the allocations card.
F	The application frees the data set before allocation. The FREE statement is generated on the allocations card.
Default:	Free

If exists (Delete, Use - for new data sets only)

Used for allocating a new, non-temporary data set. Generates a DELETE statement in the CLIST to prevent a *Duplicate Data Set Name* error.

DELETE	Generates a DELETE statement in the CLIST. This deletes any existing data set allocated with the same name.
USE	A *Duplicate Data Set Name* error is generated if there is an existing data set allocated with the same name.
Default:	USE

Suppress (Yes or No)

Indicates whether or not this panel should be suppressed when converting JCL. Once this option is set to Yes, this panel only displays when choosing option 4 from the Allocations Main Menu.

Chapter 7: JCL Conversion for CA Roscoe

This chapter describes how to use the JCL Conversion facility for CA Roscoe.

The JCL Conversion facility lets you convert JCL to CA Roscoe RPF format so that CA Roscoe can execute it. Use this facility to convert the JCL that performs the allocations required by programs you are testing in foreground with your product.

This section contains the following topics:

Access the JCL Conversion Facility

To enter the JCL Conversion facility, enter the following command on the command line:

```
[xxx.]IBCONV
```

where *xxx* is the prefix of the library in which the product is installed. Omit this prefix if the application is installed in a common execution library.

The application displays the JCL Converter panel.

Specify the JCL Member for Conversion

The JCL Converter panel lets you specify the JCL member for conversion:

```
---------------------- CA InterTest Batch JCL Converter ----------------------
 Option ===>
 Select an Option and press ENTER
             A - AWS....
             L - LIB....
             D - DSN.... USER01.LIB.CNTL(TEST*)
             J - JOB...:

             X    -  Exit


 JCL Conversion PROCLIBs:
             Proclib1... USER01.PROCLIB
             Proclib2... SYS2.PROCLIB
             Proclib3...
             Proclib4...
             Proclib5...
             Proclib6...
             Proclib7...
```

Identify Where the JCL Member Is Stored

JCL members are stored in one of the following areas:

AWS	CA Roscoe active working storage
LIB	CA Roscoe user library
DSN	Data set (PDS, sequential or CA Librarian)
JOB	The currently attached job. You cannot change this jobname.

The information specified in these fields reflects your profile. If you change this information, for example if you change the DSN, application updates your profile, which is stored in your library in member ZZIBOPTS. Follow standard CA Roscoe conventions for specifying the AWS, Library or DSN.

Identify the PROCLIBs

In addition to specifying where the JCL member is stored, you must specify at least one PROCLIB. Each PROCLIB identifies a library where JCL procedures are stored. Even if the JCL member does not use a procedure, one PROCLIB must be specified on the JCL Conversion panel.

Display the JCL Member

To display the JCL member

1. Enter a letter (A,L,D,J) in the Option field to indicate where the JCL member is stored.
2. Complete the information for the option you selected:
   - For A (AWS), enter the AWS name or leave this field blank.
   - For L (library) or D (DSN), enter the library or DSN name or a partial name using the generic characters + or *, or leave this field blank.
   - For J (job), the name of the currently attached job already appears on the JCL Conversion panel and cannot be changed.
3. Enter at least one PROCLIB name in the PROCLIB list.
4. Press Enter.

Result: If you entered a complete AWS, library or DSN name, CA Roscoe displays the JCL member you selected. If you entered a partial name or left the name blank, CA Roscoe prompts you for additional information. See Selecting the JCL Member from a List, which follows, for instructions on specifying that information.

If you selected J (job), CA Roscoe displays file 2 (the JCL file) of the currently attached job.

Select the JCL Member from a List

If you do not specify a complete AWS, library or DSN name on the JCL Converter panel, CA Roscoe prompts you for additional information.

For *AWS*, CA Roscoe displays a list of members. Enter **a** to the left of the desired AWS and press Enter to attach and display the member.

For a *library*:

- If you left the library field blank on the JCL Converter panel, CA Roscoe displays the Library Facility panel. Specify the library information and, optionally, the member name, and attach the library.
  - If you specify the member name, CA Roscoe displays the member.
  - If you omit the member name, CA Roscoe displays the member list.
- If you specified a partial library name on the JCL Converter panel, CA Roscoe displays a list of members that meet the criteria.

When CA Roscoe displays the member list, enter **a** to the left of the library JCL member and press Enter to attach and display a member.

For a *DSN*:

- If you left the DSN field blank on the JCL Converter panel, CA Roscoe displays the Data Set Facility panel. Specify the data set information and attach the data set to display a list of members.
- If you specified a partial data set name on the JCL Converter panel, CA Roscoe displays a list of members that meet the criteria.

When CA Roscoe displays the member list, enter **a** to the left of the data set JCL member and press Enter to attach and display a member.

Convert the JCL Member

When the application displays the JCL member you specified on the JCL Converter panel or selected from a list, you can review the member to make sure it is the one you want to convert.

To convert the member

1. Enter **GO** on the command line.
2. Press Enter.

Result: the application converts the JCL member to CA Roscoe RPF format and displays the following message:

`RPF generated in AWS` *xxxx*

Where *xxxx* is the current AWS.

After the JCL member has been converted, enter **a** to attach and display the converted JCL member.

Review and Edit the Converted JCL

You can review and, if desired, edit the converted JCL to delete unnecessary steps. Note the following:

- The converted JCL is divided into steps.
- **::** denotes a comment that identifies a STEP in the original JCL and displays the EXEC card.
- **:** denotes a message generated during conversion. These messages can be ignored.

- Each step contains:
  - FREE statements to deallocate previously allocated data sets.
  - DELETE statements, if necessary, to delete non-temporary data sets that need to be created and may already exist.
  - ALLOC statements to allocate and, if necessary, create the data sets.

Use standard CA Roscoe editing commands to edit the converted JCL. When you finish editing the member, save it in your own CA Roscoe library.

Execute the Converted JCL

After the JCL member has been saved in your CA Roscoe library, you can execute the converted JCL member at any time to perform the allocations. Before executing a program that you want to test with the application, execute the appropriate RPF member to allocate the files required by the program.

Convert JCL to ALIB in Batch

A utility exists, CAMRAUTL, which takes a given JCL member of a PDS and creates an ALIB member from it. Review JCL member CAMRAUTL in CAMRPROC. There are three required DDs, shown in the following table:

DDname	Description
JCL	DD pointing to the JCL member to be used in creating the ALIB.
ALIB	DD pointing to the ALIB PDS.
SYSPROC	DD pointing to the PROCLIB concatenation.

JCL example:

```
// JOB
//*********************************************************************
//CRE8ALIB EXEC PGM=CAMRAUTL
//STEPLIB  DD  DISP=SHR,DSN=CAI.CAMRLOAD
//SYSPROC  DD  DISP=SHR,DSN=USER.JCL
//JCL      DD  DISP=SHR,DSN=USER.JCL(TESTJCL)
//ALIB     DD  DISP=SHR,DSN=USER.ALIB
//
```

Chapter 8: Batch Link Facility

Batch link lets you debug applications from a terminal while they execute in batch.

The batch link facility offers the following advantages over standard foreground debugging:

- No foreground allocations

  Executing an application in foreground requires that all data sets used by the application be first allocated in foreground. While the application provides a JCL conversion and allocations facility for this purpose, the allocation and de-allocation process can complicate the debugging process. Applications debugged using batch link are executing in batch, so no foreground allocations are required.

- Execution of leading and trailing steps

  Sometimes an application is executed as part of a multi-step job stream and one or more leading steps must be executed before the application can run. Batch link allows debugging of any job step executing in batch, and therefore permits the normal completion of leading steps prior to debugging. After debugging, any trailing steps can be completed prior to job termination.

- Use of temporary files

  Because batch link permits the execution of leading steps, applications can use temporary files created by a prior step. The job stream can then delete these files when they are no longer needed.

- Multi-step debugging

  Batch link supports the debugging of multiple steps within the same job stream. When debugging of any step completes, the job continues as usual with the execution of the following step until end of job. Each batch link user controls which steps will be debugged within a job stream before the job is submitted.

- Multi-user debugging

  At any time during a batch link debugging session, you can suspend the session and disconnect the terminal from the batch debugging session. This lets the debugging session be retrieved by another (or even the same) user at a later time. This facilitates the transfer of the debugging session from one user to another without terminating and restarting the application.

This section contains the following topics:

JCL Requirements

Before debugging a batch application using the batch link facility, modify the application JCL to enable the batch link environment. Do this manually as described here, or automatically using the batch link JCL conversion facility.

Change the application JCL for each step to be debugged. These changes include the following steps:

1. Modify the PGM= field on the // EXEC statement to execute CAMRBL01.
2. Insert the application executable library at the end of the STEPLIB or JOBLIB concatenation if it does not exist in LINKLIST.

   Note: It is possible to use the dynamic symbolic support feature if you:

   - Have multiple CA Endevor SCM C1DEFLTS

   and

   - Are debugging as if you have a single C1DEFLTS.

   To do so, insert a fully qualified data set name containing the C1DEFLTS with your site under the STEPLIB concatenation ID.
3. Insert additional DD statements for the application data sets used by the debugging engine, including INT1PARM, INT1PROF, INT1PNLL, and INT1MSGL.

   Note: For more information about these additional DD statements, see The Batch Link DD Statements.
4. Insert an INT1OPTS DD containing in-stream debugging options, such as the original program name from the PGM= field on the // EXEC statement.

   Note: For more information about valid options, see Batch Link Options (see page 208).

Note: Different requirements apply for DB2 applications (see page 218).

The following code is a sample JCL stream for batch link:

```
// JOB
// EXEC PGM=CAMRBL01,PARM='application parms'
//STEPLIB  DD DISP=SHR,DSN=CAI.CAMRLOAD
//         DD DISP=SHR,DSN=USER.LOADLIB
//         DD DISP=SHR,DSN=USER.C1DEFLTS
//INT1PARM DD DISP=SHR,DSN=CAI.CAMRSAMP
//INT1PROF DD DISP=SHR,DSN=CAI.PROFLIB
//INT1PNLL DD DISP=SHR,DSN=CAI.CAMRPNL1
//INT1MSGL DD UNIT=SYSDA,SPACE=(TRK,(1,1))
//INT1OPTS DD *
  EXEC=CAMRCOBB,PROFILE=USER01
/*
```

Debug a Batch Application

When a job step that has been modified for batch link executes, the program CAMRBL01 starts the debugging engine in batch. At this point the job step becomes available for debugging.

Batch link debugging sessions can only be connected to a terminal using the Batch Link Option Menu, described later in this section. You choose a batch link job step for debugging from the selection list provided. Doing so connects the terminal to the batch job, creating a batch link debugging session.

Debugging is exactly the same as it is for a standard foreground testing session, with the following exceptions:

- The application and the debugging engine both execute in batch, not in foreground. The terminal acts only as a *window* for communicating with the batch link session.
- The batch link session begins with the Monitor Control panel, not an Execution Control panel.
- When the debugging session ends, the batch job continues normally with the next step, and the terminal returns to the Batch Link Option Menu.

Use the Batch Link Menus

You can display the Batch Link menu by selecting option 5 from the Primary Option Menu:

The Batch Link Option menu consists of four options:

JCL

Use this option to convert and submit JCL for batch link. Conversion is only required once per JCL stream, unless you alter your original JCL or decide to change which steps to debug. Alternatively, you can modify the JCL manually.

Batch Link

Use this option to select a batch job for debugging. Batch link lists all jobs available for debugging. Select the job that you want to debug to connect your terminal to the batch job, creating a batch link session.

Edit JCL

Use this option to edit and submit existing batch link JCL.

Note: This option is only supported under ISPF.

Scheduling

Use this option to use the DB2 SP and IMS/DC scheduling feature. This feature lets you specify which programs in which regions you want the application to monitor, and lets all other programs run unmonitored. This feature lets you use the same region for debugging and for running the normal IMS/DC and DB2SP workload.

The Batch Link JCL Conversion Panel

Option 1 of the Batch Link Option Menu displays the Batch Link JCL Conversion panel. Use this panel to identify the library name (and member if the library is partitioned) containing the JCL to be converted for batch link.

Note: The batch link JCL conversion utility creates a new JCL member in the output destination of your choice.

The Batch Link Conversion panel fields are described next:

Field Name	Description
Project	The high-level qualifier for the library containing the JCL to be converted.
Library	The middle-level qualifier for the library containing the JCL to be converted.
Type	The lowest-level qualifier for the library containing the JCL to be converted.
Member	The name of the member to be converted when the library is partitioned. Leave this field blank for sequential files or to produce a member list for a PDS.
Data Set Name	If the JCL library name does not conform to PROJECT.LIBRARY.TYPE, enter the name of the library in this field. Use quotes for fully qualified names. If quotes are omitted, your ISPF prefix is appended as the high-level qualifier.
Volume Serial	If the JCL library is not catalogued, use this field to enter the VOLSER of the DASD volume on which the library resides.
Dataset Password	For password-protected libraries only, enter the password in this field.

Once you have given values to all of the necessary fields on the panel, press Enter to advance to the next panel.

Select a JCL Member

If you have not provided a member name and your JCL library is partitioned, you are prompted to select a member for conversion.

```
----------------- CA InterTest Batch MEMBER SELECTION ------------------------
 COMMAND INPUT ===>                                          SCROLL ===> HALF
      NAME
      ****************************** TOP OF DATA ********************************
      ARMYCOMP
      ARMYRUN
      ARMY5STP
      ASM1
      COMPNGO
      DB2COMP
      DB2DEMO2
      DB2DEMO3
      DB2RUN
    s DEMOJCL
      IMSBUILD
      ***************************** BOTTOM OF DATA ******************************
```

To select a member, place an **s** in the prefix area next to the member name and press Enter.

The Batch Link Conversion Options Panel

After you identify the JCL library, use the Batch Link Conversion Options panel to specify options for the conversion.

The Batch Link Conversion Options panel fields are described next:

Field Name	Description
Proclib 1-7	The data set names of any procedure libraries required for converting the JCL.
Preallocated DD	Identifies the ddname used to pre-allocate procedure libraries required for converting the JCL.
Output JCL Lib	The name of the JCL library where the converted JCL should be written. If this field is left blank, a temporary output file is allocated.
	When specifying an output JCL library, use quotes for fully qualified data set names. If quotes are omitted, your ISPF prefix is appended as the high-level qualifier.

Field Name	Description
Output Member	If a partitioned output JCL library is specified, enter the member name into which the converted JCL is to be written.
Unit	The unit field to be used if the converted JCL is being written to a temporary file. **Note:** If using a temporary file for the converted JCL, VIO must not be used.
Profile Library	The name of the application profile library that should be used by the batch debugging engine, excluding the profile member name. Use quotes for fully qualified data set names. If quotes are omitted, your ISPF prefix is appended as the high-level qualifier.
Profile Member	The name of the member within the profile library that should be used by the batch debugging engine.
Options	Enter additional batch link options in this field. These options are discussed in the Batch Link Options section.

After you assign values to all of the necessary fields on the panel, press Enter to advance to the next panel.

Select Job Steps

If your JCL contains multiple steps, you are prompted to select job steps.

```
----------------- CA InterTest Batch Step Selection List ----------------------
 COMMAND INPUT ===>                                            SCROLL ===> PAGE
      Select one or more of the following Steps:
      ***************************** TOP OF DATA ********************************
 s    1     //STEP1    EXEC PGM=CAMRCOBB,REGION=4M
      2     //STEP2    EXEC PGM=CAMRCOB,REGION=4M
      ************************** BOTTOM OF DATA ******************************
```

You can select any or all job steps in the selection list. To select a job step, place an **s** in the prefix area next to the entry.

To select multiple job steps, place an **s** in the prefix area next to each entry.

Press Enter to convert the JCL for batch link.

Convert and Submit the JCL

Once the JCL is converted, you are placed into EDIT, which lets you view or further modify the JCL prior to submitting it.

If you entered an output data set name, the JCL is written to that data set. If no output data set name is provided, a temporary sequential file is used.

```
EDIT       SYS00222.T093343.RA000.USER01.IBJCL.H01         Columns 00001 00072
Command ===>                                                 Scroll ===> CSR
****** ***************************** Top of Data ****************************
000001 //USER01A  JOB (42400000),'Demo JCL',CLASS=K,MSGCLASS=X,
000002 // NOTIFY=USER01
000003 //STEP1    EXEC PGM=CAMRBL01,REGION=4M
000004 //STEPLIB  DD  DISP=SHR,DSN=CAI.CAMRLOAD
000005 //         DD  DISP=SHR,DSN=USER01.LOADLIB
000006 //SYSPRINT DD  SYSOUT=*
000007 //SYSOUT   DD  SYSOUT=*
000008 //INT1OPTS DD *
000009   EXEC=CAMRCOBB, PROFILE=USER01
000010 /*
000011 //INT1PARM  DD  DISP=SHR,DSN=USER01.INTBATCH.CAMRSAMP
000012 //INT1LOAD  DD  DISP=SHR,DSN=USER01.INTBATCH.CAMRLOAD
000013 //INT1PNLL  DD  DISP=SHR,DSN=USER01.INTBATCH.CAMRPNL1
000014 //INT1MSGL  DD  DISP=SHR,DSN=USER01.INTBATCH.CAMRMSG0
000015 //INT1PROF  DD  DISP=SHR,DSN=USER01.INTBATCH.PROFLIB
000016 //INT1CLIB  DD  DISP=SHR,DSN=USER01.INTBATCH.CAMRSRC
000017 //INT1CLOG  DD  SYSOUT=*
000018 //INT1REPT  DD  SYSOUT=*
****** ************************** Bottom of Data ***************************
```

For each job step selected for debugging, the following modifications are made for batch link:

1. The PGM= field on the // EXEC statement is modified to execute CAMRBL01, the batch link driver program.
2. The application executable library is added to the STEPLIB or JOBLIB concatenation.
3. Additional DD statements are inserted for the application data sets used by the debugging engine, including INT1PARM, INT1PROF, INT1PNLL, and INT1MSGL. These additional DD statements are described in detail later.
4. An INT1OPTS DD statement is added containing in-stream debugging options, such as the original program name from the PGM= field on the // EXEC statement.
5. If you are using the dynamic symbolic support feature and you have multiple C1DEFLTS, you can concatenate to the STEPLIB a fully qualified data set name containing the CA Endevor SCM site ID associated with the load module being monitored. Alternatively, you can also let CA CCI discover the site id and populate the symbolic file for you.

Note: The requirements are slightly different for DB2 applications, as described.

When you are ready to submit the JCL, enter SUBMIT on the command line and press Enter. The job is submitted but you remain in the EDIT session. Exit the EDIT session by entering END or CANCEL on the command line and then pressing Enter.

The job begins execution and becomes available for debugging as soon as one of the selected steps begins to execute.

JCL Conversion Limitations

The following JCL constructs are not currently supported by the Batch Link JCL Conversion utility:

Partial DD overrides

Given the following JCL:

```
//PROC1    PROC
//S1       EXEC PGM=TESTPGM
//TEST1    DD   DISP=(,CATLG,DELETE),DSN=USER01.TESTDSN,
//         UNIT=DISK,SPACE=(1,(1,1))
// PEND
//MYPGM    EXEC PROC1
//S1.TEST1 DD   UNIT=SYSDA
```

The DD TEST1 will not be generated correctly in the converted JCL.

JCL invokes a PROC that updates or adds JCL keywords to an EXEC statement.

Given the following JCL:

```
//PROC1    PROC
//S1       EXEC PGM=TESTPGM,PARM='ABC'
//TEST1    DD   SYSOUT=*
// PEND
//MYPGM    EXEC PROC1,PARM='DEF'
```

The PARM keyword on the EXEC card will not be updated.

Note: Any client hitting this limitation can still use the utility to do most of the conversion or you can edit the JCL and make the necessary changes manually.

When the application converts JCL for batch link, all the called catalogued procedures are expanded and all expanded steps from the procedures are given a stepname of just procstepname.

When there are COND= parameters in the original JCL with *stepname.procstepname* references, these are modified to use the *procstepname*. As a result, the reference may no longer be unique.

IF statements and DD statements with *stepname.procstepname* references are not modified and the *stepname.procstepname* reference is no longer valid. Therefore you must examine the converted JCL and modify these instances as appropriate.

The Batch Link Selection Panel

Option 2 of the Batch Link Option Menu displays the Batch Link Selection panel. Use this panel to select a batch job for debugging.

Note: This panel is not accessible unless the batch link facility is initialized on your system and at least one batch link job step is ready for debugging.

A batch link debugging session is functionally identical to a foreground test session.

Note: For a detailed description of the foreground test facility, see the chapter Debugging Commands (see page 57).

```
----------------- CA InterTest Batch Batch Link Selection --------------------
COMMAND INPUT ===>                                          SCROLL ===> PAGE
           *         *         *         *         *         *         *
     #     Jobname   Stepname  Program   Owner     User      Trans     SYSID
     ***************************** TOP OF DATA ***********************************
     1     JOB01A    STEP1     TESTPGM   USER01                        TSO01
     2     JOB02A    STEP1     CAMRCOBB  USER02                        TSO02
     3     JOB03X    RUN       PAYROLL   USER03   LINKED BY USER03     TSO01
     ************************ BOTTOM OF DATA ***********************************
```

Debugging a Job Step

To select a job step for debugging, type **S** next to the entry on the selection list, and press Enter. The Monitor Control panel appears and the debugging session begins.

Note: Only one user at a time can select a batch link job step. If another user has selected a step, the entry is marked as 'Linked by *userid*'.

Suspended Jobs

When you select a batch link job step that was suspended, the debugging session resumes at the point at which the session was suspended.

To suspend a session enter the SUSPEND command after the initial debugging intercept. A session is suspended if a user ID is canceled during a debugging session.

Debugging on Foreign LPARs

You can debug a job that is executing on a different LPAR than the one you are currently logged on to. The SYSID column displays the LPAR the job is executing on.

After you select a job that is executing on a foreign LPAR, enter your user credentials to start debugging.

The following prerequisites apply to debugging a job on a foreign LPAR:

- The LPAR must be located within the SYSPLEX.
- TCP/IP must be active and accessible across the SYSPLEX.
- A CA Testing Tools Server must be running on the LPAR.
- The SYSPLEX option available in the CAMRSTBZ program must be enabled during configuration.

  Note: For more information, see the Enable the SYSPLEX (Cross-LPAR) Debugging Feature section in the *CA InterTest Batch Installation Guide*.

The Batch Link JCL Edit Panel

Option 3 of the Batch Link Option Menu displays the Batch Link Edit JCL panel. Use this panel to identify the library name (and member if the library is PDS) containing the JCL to be edited before submitting it for the batch link debugging session.

The following table describes the Batch Link JCL Edit panel fields.

Field Name	Description
Project	The high-level qualifier for the library containing the JCL to be edited.
Library	The middle-level qualifier for the library containing the JCL to be edited.
Type	The lowest-level qualifier for the library containing the JCL to be edited.
Member	The name of the member to be edited when the library is partitioned. Leave this field blank for sequential files or to produce a member list for a PDS.
Data Set Name	If the JCL library name does not conform to PROJECT.LIBRARY.TYPE, enter the name of the library in this field. Use quotes for fully qualified names. If quotes are omitted, your ISPF prefix is appended as the high-level qualifier.

Field Name	Description
Volume Serial	If the JCL library is not catalogued, use this field to enter the VOLSER of the DASD volume on which the library resides.

Once you have given values to all of the necessary fields on the panel, press Enter to advance to the ISPF Edit panel. After editing, you can save your changes and submit the changed JCL for batch link execution.

Select a JCL Member for Editing

If you have not provided a member name and your JCL library is partitioned, you are prompted to select a member for editing.

```
----------------- CA InterTest Batch MEMBER SELECTION ------------------------
COMMAND INPUT ===>                                          SCROLL ===> HALF
      NAME
      ***************************** TOP OF DATA ********************************
      ARMYCOMP
      ARMYRUN
      ARMY5STP
      ASM1
      COMPNGO
      DB2COMP
      DB2DEMO2
      DB2DEMO3
      DB2RUN
    s DEMOJCL
      IMSBUILD
      ************************** BOTTOM OF DATA ******************************
```

To select a member, place an **s** in the prefix area next to the member name and press Enter.

The Batch Link DD Statements

The following table describes the additional DD statements required for batch link. You must add these DD statements to each job step selected for debugging:

DDNAME	Data Set Name	Description
INT1PARM	CAI.CAMRSAMP	Contains the parameter members used by the batch debugger.
INT1LOAD	CAI.CAMRLOAD	Contains the product executables.
INT1PNLL	CAI.CAMRPNL1	Contains panel skeletons.
INT1MSGL	CAI.CAMRMSG0	Contains messages.
INT1PROF	userid.PROFLIB	Contains profile members used by the debugger to store user preferences.

DDNAME	Data Set Name	Description
INT1OPTS	n/a	Contains batch link options and control information.
(Optional DDNAMEs)		
INT1CLIB	CAI.CAMRSRC	Contains INCLUDE members.
INT1CLOG	(SYSOUT)	Log output from the debugger.
INT1REPT	(SYSOUT)	Report output from the HIST and XSUM commands.

Batch Link Options

The following table describes the batch link options that you can specify in the INT1OPTS file for job steps selected for debugging.

Keyword	Description
AUTH	Specifies the USERID that is authorized to debug this application.
BYPASS	Use this option to prevent the batch link engine from stopping for input at the Monitor Control panel, the Initial Intercept panel, or both. Specify BYPASS=MONITOR to prevent the batch link engine from stopping at the Monitor Control panel unless an error is detected. **Note:** The dynamic symbolic support feature is not supported when BYAPSS=MONITOR is specified in conjunction with PGM and PROTSYM options. Specify BYPASS=INITIAL to prevent the batch link engine from stopping at the initial Intercept panel.
DB2SP	Use this option to indicate that the application you are debugging is a DB2 stored procedure.
EXEC*	Specifies the name of the application's main program. For most batch job steps, this is the name which is replaced by CAMRBL01 in the PGM= parameter of the // EXEC statement.
	For some special environments such as DB2, see Special Considerations later.

Keyword	Description
ICMDS	Indicates the member in your INT1CLIB DD that includes application commands to be executed at the initial intercept. **Note:** If this command is used in conjunction with BYPASS(INITIAL), these commands are not executed until a breakpoint is reached. If you would like to bypass the initial intercept and would also like a set of commands to be executed at the initial intercept, do not use BYPASS(INITIAL). Instead, have GO as the last command executed at the initial intercept.
IMSDC	Use this option to indicate that the application you are debugging is an online IMS application.
LOOP	Specifies the interval in seconds that a batch link job step is permitted to execute between responses from a terminal. If this interval is exceeded, the debugger cancels the batch link job step.
PGM	Use the PGM option to specify which programs are to be debugged during the batch link session. Values specified for this option are used to complete the PROGRAM fields on the Monitor Control panel for the batch link session. You can specify only one program as a value, but the PGM option can be repeated up to 30 times per job step. If there are no PGM values specified in INT1OPTS, all of the symbolic information is retrieved from the profile member identified by the PROFILE option. However, if you specify PGM values in INT1OPTS, then no values are retrieved from the profile. **Note:** If specified in conjunction with BYPASS=MONITOR and PROTSYM option, dynamic symbolic support is not supported.
PROFILE	Specifies the name of the profile member that should be used by the debugging engine. You can use a unique profile member for each batch link job or step if desired, or specify your own user ID to use your personal profile member. If not specified, the default is the user ID used to submit the job.
PROTSYM	Use the PROTSYM option to specify the file that contains the symbolic information for the programs you want to debug. You can specify only one PROTSYM file as a value, but the PROTSYM option can be repeated up to seven times.

Keyword	Description
	If the PGM option has not been specified, the PROTSYM option is ignored. **Note:** If specified in conjunction with BYPASS=MONITOR and PGM option, dynamic symbolic support is not supported.

Note: An asterisk (*) next to a keyword indicates that the option is required on each selected step.

Specify options using the syntax:

```
keyword=value
```

You can specify more than one option per line if desired, separating the options by a space or a comma.

The DB2 and IMS Schedule Menu

Option 4 of the Batch Link Option Menu displays the DB2 and IMS Schedule Menu panel:

DISPLAY

Displays a list (see page 212) of existing DB2 or IMS schedules. Specify **DB2** or **IMS** in the Schedule Monitoring for field when you use this option.

Note: You can also use these lists to change (see page 214) or delete (see page 216) entries from the existing schedule.

ADD

Allows adding a DB2 SP or IMS/DC schedule. Specify **DB2** or **IMS** in the Schedule Monitoring for field when you use this option.

IMPORT

Enables importing a predefined set of schedules (DB2 SPS and IMS/DC) from an external data set.

EXPORT

Enables exporting all schedules (DB2 SPS and IMS/DC) to an external data set.

Sharing Schedules between LPARs in a SYSPLEX

Scheduling is not limited to the LPAR you are currently logged on to. You can:

- Add an IMS / DB2 schedule on one LPAR which is then available on all LPARs.
- Change an IMS / DB2 schedule on one LPAR, the change propagates to all LPARs.
- Delete an IMS / DB2 schedule from one LPAR which is then deleted on all LPARs.
- Trigger an IMS /DB2 schedule and select it for debugging from any LPAR.
- Use the IMPORT and EXPORT functions for all scheduled items available on the LPARs within the SYSPLEX.

To use this functionality, ensure that:

- TCP/IP is active and accessible across the SYSPLEX.
- A CA Testing Tools Server is running on each LPAR that is sharing scheduling.
- You enabled the SYSPLEX option available in the CAMRSTBZ program during configuration.

  Note: For more information, see the Enable the SYSPLEX (Cross-LPAR) Debugging Feature section in the *CA InterTest Batch Installation Guide*.

The Batch Link Schedule Display Panel

Based on the database type selected in the Schedule Options panel, the following active schedule entries are displayed:

```
--------------- CA InterTest Batch DB2 SPS Schedule Maintenance --------------
COMMAND ===>                                                  SCROLL ===> CUR
   COMMAND      : A - ADD a new entry to Schedule.
   LINE COMMANDS: D - DELETE selected entry. C - CHANGE selected entry
------------------------------------------------------------------------------
CMD    REGION/JOBNAME   PROGRAM       TYPE     ONE TIME   USER       TRANSACTION
     ************************** TOP OF DATA ********************************
       BIMNTH01         ASSET*        DB2      YES
       BIMNTH01         SALES*        DB2      YES
       WKLYPAY          TIMEATTN      DB2      YES
       WKLYPAY          FCIA          DB2      YES
       WKLYPAY          FEDW2         DB2      YES
       WKLYPAY          STATEW2       DB2      YES
       WKLYPAY          CASHFLOW      DB2      YES
       QTRLY*           GL001         DB2      YES
       QTRYL*           ASSET001      DB2      YES
       QTRLY*           SALES001      DB2      YES
       QTRLY*           INVNTROY      DB2      YES
       QTRLY*           CHAININV      DB2      YES
       QTRLY*           WHAREHSE      DB2      YES
       QTRLY070         DSTRUTON      DB2      YES
     ************************** BOTTOM OF DATA *******************************
```

The Batch Link IMS/DC Schedule Display panel displays IMS/DC on the first line of the panel, and IMS under the TYPE column. The rest of the display is similar to the DB2 SPS display.

```
--------------- CA InterTest™ IMS/DC  Schedule Maintenance --------------
COMMAND ===>                                                  SCROLL ===> CUR
   COMMAND      : A - ADD a new entry to Schedule.
   LINE COMMANDS: D - DELETE selected entry. C - CHANGE selected entry
------------------------------------------------------------------------------
CMD    REGION/JOBNAME   PROGRAM   TYPE   ONE TIME   USER       TRANSACTION
     ************************** TOP OF DATA ********************************
       BIMNTH01         ASSET*    IMS    YES        IMSUSER1   IMSTRANS
       BIMNTH01         SALES*    IMS    YES        USER2IMS   IMSTRX02
       WKLYPAY          TIMEATTN  IMS    YES        IMSU0003   IMST0003
       WKLYPAY          FCIA      IMS    YES        MPPUSER4   MPPTRX04
       WKLYPAY          FEDW2     IMS    YES        USERIMS5   TESTTRAN
       WKLYPAY          STATEW2   IMS    YES        USERMPP6   TRANSMPP
     ************************** BOTTOM OF DATA *******************************
```

The Batch Link Schedule Add Panel

Select option 2 from the Schedule Options panel or enter **A** on the Schedule Display panel. The Schedule Add panel displays.

The Batch Link Schedule Add panel fields are described next:

Field Name	Description
REGION/JOBNAME	This is usually required for IMS/DC scheduling. If program name selection is independent of region, job name, or both, leave this field blank, otherwise specify a maximum of eight character IMS region name or z/OS job name that is commonly associated with the program name to be monitored. Use the entire length of the field, including trailing blanks as selection criteria. A trailing asterisk (*) denotes a generic name. Characters to the left of the asterisk are used for selection. Thus ABC* matches any name whose first three characters start with ABC. An asterisk (*) as the first character of this field denotes that any region/job name is considered a match.
PROGRAM NAME	This is a required field. Provide a maximum of eight-character valid program name (DB2 SP name or IMS/DC name) you want the application to monitor during your debugging session. The entire length of the field, including trailing blanks is used as selection criteria. A trailing asterisk (*) denotes a generic name. Characters to the left of the asterisk are used for selection. Thus ABC* matches any name whose first three characters start with ABC. An asterisk (*) as the first character of this field denotes that any program name is considered a match.
ONE TIME SCHEDULE	If this option is set to **Y**, yes, the entry is deleted from the schedule upon selection for monitoring. Change this to **N**, no, if you do not want the entry deleted after selection. Note that keeping the entry in the schedule means that the product always monitors the named program in a debugging session when all of the selection criteria are met.
DATABASE TYPE	This field is automatically filled in based on the value you entered in Schedule Monitoring for ==> of the Schedule Options panel.

Field Name	Description
USER ID	This field is used for IMS/DC only. This is the user ID assigned to the transaction submitted in the MPP region. If this value is supplied, monitoring happens only if the specified user ID starts the transaction.
TRANSACTION	This field is the IMS transaction name used for IMS/DC only. A trailing asterisk (*) denotes a generic name. Characters to the left of the asterisk are used for selection. For example ABC* matches any name whose first three characters start with ABC.

The Batch Link Schedule Change Panel

To change displayed entries, enter **C** under the CMD column next to the entries you want to change:

```
--------------- CA InterTest Batch DB2 SPS Schedule Maintenance --------------
COMMAND ===>                                                  SCROLL ===> CUR
   COMMAND      : A - ADD a new entry to Schedule.
   LINE COMMANDS: D - DELETE selected entry. C - CHANGE selected entry
------------------------------------------------------------------------------
CMD    REGION/JOBNAME   PROGRAM       TYPE     ONE TIME   USER       TRANSACTION
     ***************************** TOP OF DATA ***********************************
       BIMNTH01         ASSET*        DB2      YES
       BIMNTH01         SALES*        DB2      YES
C      WKLYPAY          TIMEATTN      DB2      YES
       WKLYPAY          FCIA          DB2      YES
       WKLYPAY          FEDW2         DB2      YES
       WKLYPAY          STATEW2       DB2      YES
       WKLYPAY          CASHFLOW      DB2      YES
C      QTRLY*           GL001         DB2      YES
C      QTRYL*           ASSET001      DB2      YES
       QTRLY*           SALES001      DB2      YES
       QTRLY*           INVNTROY      DB2      YES
       QTRLY*           CHAININV      DB2      YES
       QTRLY*           WHAREHSE      DB2      YES
       QTRLY070         DSTRUTON      DB2      YES
     **************************** BOTTOM OF DATA ********************************
```

The Schedule Change panel displays once for each selection:

```
------------------ CA InterTest Batch DB2 SPS Schedule Change ----------------

                 CHANGE any field by overtyping the displayed value.
                 Press ENTER to accept the change, or END to cancel.

--------------------------------------------------------------------------------
               REGION/JOBNAME      ====>  WKLYPAY
               PROGRAM NAME        ====>  TIMEATTN
               ONE TIME SCHEDULE   ====>  Y        (Y/N)
               DATABASE TYPE       ====>  DB2
               USER ID             ====>
               TRANSACTION         ====>


     PFkeys: HELP - Tutorial
             END  - Exit
```

After changing the schedule and pressing Enter, the Change panel for the next marked line, if any, is displayed. This continues until all of the lines marked with C have been displayed for change.

After all the changes have been done, the system refreshes the Display panel to reflect the changes.

Note: Pressing the END key (PF3) only cancels the currently displayed change and any further Change panels queued for display.

For more information about the Change panel fields, see The Batch Link Schedule Add Panel section.

The Batch Link Schedule Delete Action

To delete an entry, enter **D** under the CMD column next to the entry you want to delete and press Enter:

```
--------------- CA InterTest Batch DB2 SPS Schedule Maintenance --------------
COMMAND ===>                                                  SCROLL ===> CUR
   COMMAND      : A - ADD a new entry to Schedule.
   LINE COMMANDS: D - DELETE selected entry. C - CHANGE selected entry
------------------------------------------------------------------------------
CMD    REGION/JOBNAME   PROGRAM       TYPE     ONE TIME   USER       TRANSACTION
     ***************************** TOP OF DATA ********************************
       BIMNTH01         ASSET*        DB2      YES
       BIMNTH01         SALES*        DB2      YES
D      WKLYPAY          TIMEATTN      DB2      YES
       WKLYPAY          FCIA          DB2      YES
       WKLYPAY          FEDW2         DB2      YES
       WKLYPAY          STATEW2       DB2      YES
       WKLYPAY          CASHFLOW      DB2      YES
D      QTRLY*           GL001         DB2      YES
D      QTRYL*           ASSET001      DB2      YES
       QTRLY*           SALES001      DB2      YES
       QTRLY*           INVNTROY      DB2      YES
       QTRLY*           CHAININV      DB2      YES
       QTRLY*           WHAREHSE      DB2      YES
       QTRLY070         DSTRUTON      DB2      YES
     **************************** BOTTOM OF DATA ******************************
```

The system deletes all the marked entries and refreshes the display to reflect the deletion.

The Batch Link Schedule Import Panel

When you select option 3 from the Schedule Options panel, the Batch Link Schedule Import panel displays.

The Batch Link Schedule Import panel fields are described next:

Field Name	Description
Project	The high-level qualifier for the data set containing the schedule to be imported.
Library	The middle-level qualifier for the data set containing the schedule to be imported
Type	The lowest-level qualifier for the data set containing the schedule to be imported.
Member	The name of the member to be imported when the data set is partitioned. Leave this field blank for sequential files or to produce a member selection list for a PDS.

Field Name	Description
Data Set Name	If the import data set name does not conform to PROJECT.LIBRARY.TYPE, enter the name of the data set in this field. Use quotes for fully qualified names. If quotes are omitted, your ISPF prefix is used to form the high-level qualifier.
Volume Serial	If the data set is not catalogued, use this field to enter the VOLSER of the DASD volume on which the data set resides.

Note that the DCB attributes of the IMPORT data set are RECFM=FB, LRECL=80, DSORG=PS or PO.

Alternatively you can run the schedule import function as a batch job. Use the following sample JCL to accomplish the task:

```
//  JOB
/*JOBPARM SYSAFF=xxxx
//IMPORT   EXEC PGM=CAMR40IP
//STEPLIB  DD DISP=SHR,DSN=CAI.CAMRLOAD
//INT1SKUT DD DISP=SHR,DSN=CAI.SCHEDULE.DATA(MEMBER)
```

Instead of doing a complete replacement of the schedule table, import can also perform an additive import when you code an execution parameter:

```
//  JOB
/*JOBPARM SYSAFF=xxxx
//IMPORT   EXEC PGM=CAMR40IP,PARM='ADD'
//STEPLIB  DD DISP=SHR,DSN=CAI.CAMRLOAD
//INT1SKUT DD DISP=SHR,DSN=CAI.SCHEDULE.DATA(MEMBER)
```

Entries are added based on slot availability and using the same first available rules.

The following code is return code from import operations:

RC = 0 successfully imported

RC = 4 schedule table does not exist

RC = 8 INT1SKUT DD statement missing.

The Batch Link Schedule Export Panel

When you select option 4 from the Schedule Options panel, the Batch Link Schedule Export panel displays.

See the Batch Link Schedule Import panel fields for a discussion of the various similarly named fields.

Note: Export creates a new PDS member if member name provided does not exist. If one does exist, it is overwritten.

Alternatively, you can run the schedule export function as a batch job. Use the following sample JCL to accomplish the task:

```
//  JOB
/*JOBPARM SYSAFF=xxxx
//EXPORT   EXEC PGM=CAMR40XP
//STEPLIB  DD DISP=SHR,DSN=CAI.CAMRLOAD
//INT1SKUT DD DISP=SHR,DSN=CAI.SCHEDULE.DATA(MEMBER)
```

The following code is return code from export operations:

RC = 0 successfully imported

RC = 4 schedule table does not exist

RC = 8 INT1SKUT DD statement missing.

Special Considerations

This section describes additional considerations for using batch link in special environments.

DB2 Considerations

Use batch link to debug DB2 applications, provided that the following JCL conversion rules are implemented:

- The main program name (IKJEFT01) is not replaced by CAMRBL01 in the PGM= parameter of the // EXEC statement. If you are using the Call Attach Facility of DB2, your main program name is the application name. There are no special JCL conversion rules for this environment.
- SYSTSIN is modified to identify CAMRBL01 as the main program to be invoked by DB2.

- The EXEC option in INT1OPTS must identify the name of the application's main program, the name that is replaced by CAMRBL01 in the SYSTSIN file.
- If a LIBRARY parameter exists in the SYSTSIN file, remove it. You must add to the STEPLIB or JOBLIB concatenation the library containing the application's main program and the library containing the batch link executables.

Convert DB2 JCL using the Batch Link JCL Conversion dialog.

DB2 JCL Example

The following example shows a DB2 job step that has been converted for batch link:

```
//RUN      EXEC PGM=IKJEFT01,COND=(4,LT),DYNAMNBR=20
//STEPLIB  DD DISP=SHR,DSN=CAI.CAMRLOAD
//         DD DISP=SHR,DSN=USER01.LOADLIB
//         DD DISP=SHR,DSN=SYS2.DB2510.MAINT.SDSNLOAD
//SYSTSPRT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
/*
//SYSTSIN   DD *
  DSN  SYSTEM(D510)
  RUN PROGRAM(CAMRBL01) PLAN(DB2DEMO)
  PARM('ABEND0C7/TRAP(OFF)')
  END
/*
//INT1OPTS DD *
  EXEC=DB2DEMO,PROFILE=USER01
/*
//INT1PARM  DD  DISP=SHR,DSN=CAI.CAMRSAMP
//INT1LOAD  DD  DISP=SHR,DSN=CAI.CAMRLOAD
//INT1PNLL  DD  DISP=SHR,DSN=CAI.CAMRPNL1
//INT1MSGL  DD  DISP=SHR,DSN=CAI.CAMRMSG0
//INT1PROF  DD  DISP=SHR,DSN=CAI.PROFLIB
//INT1CLIB  DD  DISP=SHR,DSN=CAI.CAMRSRC
//INT1CLOG  DD  SYSOUT=*
//INT1REPT  DD  SYSOUT=*
```

Debug Your DB2 Stored Procedures

Use batch link to debug DB2 stored procedures. Confirm with your systems programmer that the necessary changes have been made to the WLM environment to support debugging DB2 stored procedures. Also, confirm that if the scheduling feature is enabled, described in this chapter, that it is set up to debug your stored procedure.

DB2 SP SCHEDULING CONSIDERATIONS

Since DB2 SP currently runs under the WLM started task, the selection request for DB2 SP is made without the REGION/JOBNAME attribute. Thus REGION/JOBNAME Selection Phase is bypassed. You can continue to specify a REGION/JOBNAME field for DB2 SP schedule entries or you can leave that field blank.

Debug Your Online IMS Programs

Use batch link to debug your online IMS applications without the use of BTS. Confirm with your systems programmer that the DC transaction that you want to debug has its DC region set up for batch link testing. Also, confirm that if the scheduling feature, described in this chapter, is enabled, it is set up to debug your IMS transaction.

Use Batch Link Schedule to Enhance your DB2 SP or IMS/DC Debugging Experience

The batch link schedule feature helps you leverage your testing environment by letting you specify exactly which stored procedures or IMS/DC transactions you want the application to monitor. All other programs run unmonitored. This lets you use the same region for debugging as well as executing normal DB2 SP or IMS/DC workload.

Confirm with your systems programmer that the product has been initialized with this feature enabled. You can also confirm this by going into the product and choosing option 5, Batch Link. From there, choose option 4, Scheduling. From there, choose Option 1, supply a database type, and press Enter. If a short message of 'SCHEDULE TABLE NOT ANCHORED' displays, this feature has not been enabled. In this case, all DB2 SPs and IMS/DC transactions will run unmonitored.

Select DB2 SPs or IMS/DCs for monitoring

Entries in the schedule are organized top down and are tracked and managed by slot numbers (slot #1 through slot #*nnnn*). Entries are added using the first available scheme. Thus when an entry in the low slot number range (for example #2), is deleted, the next add is added to #2, using the first available scheme:

Slot Number	Region/Job Name	Program Name	TYPE	One Time
1 (top)	MON	SP*	DB2	Y
2	MON*	SP*	IMS	Y
3		SP*	DB2	Y
4	MONTH*	SP*	DB2	N

Slot Number	Region/Job Name	Program Name	TYPE	One Time
60 (bottom)				

Selection of an entry for monitoring is on a **first-and-best** match basis and in a top down fashion (slot #1 through slot #60). Select an entry for monitoring only after it has passed all the Selection Phases.

1. Enter Selection Phase - Match entries first against TYPE. Proceed to next phase if there is a match, otherwise examine the next entry.
2. REGION/JOBNAME Selection Phase - next REGION/JOBNAME of the entry having a matching TYPE is compared. If REGION/JOBNAME starts with an asterisk (*) or contains all blanks, any region or job name selection request would be treated as a match to the REGION/JOBNAME. Trailing blanks are treated as part of the name, thus the entire length (8 bytes) of REGION/JOBNAME is used for comparison (MON). Trailing asterisk makes the name generic, thus the characters and length to the left of the asterisk are used for comparison (MON).

   Proceed to the next phase if there is a match, otherwise examine the next entry.
3. Program Name Selection Phase - When the system finds a matching REGION/JOBNAME entry, it now matches its Program Name against the selection request. Trailing blanks are treated as part of the program name, thus the 8 byte Program Name including blanks is used for comparison. A trailing asterisk makes the program name generic, thus the characters and length to the left of the asterisk are used for comparison. When the program name matches that of the selection request, the program is monitored.

We will use the table (on the previous page) to illustrate monitor selection logic and flow. Assuming we have two batch link jobs:

1. JOBNAME = MONDAY; PROGRAM NAME = SPECIAL1; TYPE = DB2
2. JOBNAME = MONTH3; PROGRAM NAME = SPECIAL; TYPE = DB2

Job 1 is ready to be monitored. Attributes in the first batch link job are used as selection criteria:

1. Slot #1 Type Selection Phase ok. Match on TYPE DB2
2. Slot #1 REGION/JOBNAME Selection failed. MON (with 5 trailing blanks) does not match MONTHLY.
3. Slot #2 Type Selection Phase failed. No match on TYPE of DB2.
4. Slot #3 Type Selection Phase ok. Match on TYPE DB2
5. Slot #3 REGION/JOBNAME Selection ok. Match on REGION/JOBNAME (all blanks treated as a match)
6. Slot #3 Program Name Selection Phase ok. SP of SP* matches SP of SPECIAL1.

7. Slot #3 is deleted (ONE TIME set to Y)
8. PROGRAM SPECIAL1 is selected for monitoring.

Job 2 is now ready to be monitored. Attributes in the second batch link job are used as selection criteria:

1. Slot #1 Type Selection Phase ok. Match on DB2
2. Slot #1 REGION/JOBNAME Selection failed. MON (with 5 trailing blanks) does not match MONTHLY.
3. Slot #2 Type Selection Phase failed. No match on DB2.
4. Slot #3 deleted and is skipped.
5. Slot #4 Type Selection Phase ok. Match on TYPE DB2.
6. Slot #4 REGION/JOBNAME Selection Phase ok. Generic name MON of MONTH* matches MON of MONTH3
7. Slot #4 Program Name Selection Phase ok. Generic SP of SP* matches SP of SPECIAL.
8. Slot #4 is deleted
9. Slot #4 is selected for monitoring.

Assuming Job 2 is submitted before Job 1, Slot #3 is again selected for monitoring. However because its One Time Schedule is set to Y, Slot #3 is deleted before Job 1's selection request is made. In this case, Job 1 runs unmonitored, since MONTH of MONTH* (slot #4) does not match MONDAY (Job 1's JOBNAME).

Import and Export Schedule Files

Importing a schedule provides you with an easy way to maintain a persistent schedule across IPLs. You can also make significant scheduling changes instead of changing individual schedules manually using the online change option.

You can create the schedule import file manually when you first use the scheduling feature, or you can create it later using the schedule export option, since you can activate the scheduling feature with an empty schedule and later add individual schedules (using the online Schedule Add option) as the need arises.

The schedule export option is an easy way to create a backup copy of your fine-tuned schedule or a uniquely tailored application system schedule to facilitate systems testing.

Special Processing for DB2/IMS Schedule Export and Import

If a DB2/IMS schedule import file has the JOBNAME or region name specified as * (wildcard for entire name), that line is not imported. A * in column 1 was considered as a comment on import. This solution changes the * for jobname to ? on export and then on import changes the ? back to a *.

Sample Schedule Import File

The following example shows a sample import file and its supported record layout:

```
*************************************************************************
* CA BATCH LINK SCHEDULES SAMPLE IMPORT
* ASTERISK IN COLUMN 1 DENOTES A COMMENT LINE.
* THE SYSTEM WILL VALIDATE DATABASE TYPE FIELD, AND ONE TIME SCHEDULE FI ELD.
* VALID ONES ARE: DB2 OR IMS FOR DATABASE TYPE; Y OR N FOR ONE TIME SCHEDULE.
* OTHER FIELDS ARE ACCEPTED ASIS, WITHOUT VALIDATION.
* FORMAT OF IMPORT/EXPORT RECORD:
* COLUMN 1  - 8  REGION/JOBNAME (8 BYTES)
* COLUMN 10 - 17 PROGRAM NAME   (8 BYTES)
* COLUMN 19 - 19 ONE TIME SCHEDULE (1 BYTE. Y OR N)
* COLUMN 21 - 23 DATA BASE TYPE (3 BYTES. DB2 OR IMS)
*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
BIMNTH01 GL001    Y DB2
BIMNTH01 ASSET*   Y DB2
BIMNTH01 SALES*   Y DB2
WKLYPAY  TIMEATTN Y DB2
QTRLY080 OXLYSABN Y IMS
QTRLY090 SAP001   Y IMS
QTRLY100 SAP002   Y IMS
```

Sample Schedule Export File

The following example shows a sample export file with a system generated record layout:

```
*************************************************************************
* CA BATCH LINK SCHEDULES SAMPLE EXPORTED FROM SID: CA31 ON 07/14/2014 AT 09:41
* ASTERISK IN COLUMN 1 DENOTES A COMMENT LINE.
* THE SYSTEM WILL VALIDATE DATABASE TYPE FIELD, AND ONE TIME SCHEDULE FI ELD.
* VALID ONES ARE: DB2 OR IMS FOR DATABASE TYPE; Y OR N FOR ONE TIME SCHEDULE.
* OTHER FIELDS ARE ACCEPTED ASIS, WITHOUT VALIDATION.
* FORMAT OF IMPORT/EXPORT RECORD:
* COLUMN 1  - 8  REGION/JOBNAME (8 BYTES)
* COLUMN 10 - 17 PROGRAM NAME   (8 BYTES)
* COLUMN 19 - 19 ONE TIME SCHEDULE (1 BYTE. Y OR N)
* COLUMN 21 - 23 DATA BASE TYPE (3 BYTES. DB2 OR IMS)
*---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
BIMNTH01 GL001    Y DB2
BIMNTH01 ASSET*   Y DB2
BIMNTH01 SALES*   Y DB2
WKLYPAY  TIMEATTN Y DB2
QTRLY080 OXLYSABN Y IMS
QTRLY090 SAP001   Y IMS
QTRLY100 SAP002   Y IMS
```

An asterisk is used as a wild card character to match all or part of a name. An asterisk in the first column of a name indicates that any name is matched. However, if any region/jobname will be matched, that is - the entire region/jobname is specified as a wildcard, then column 1 must be a ? in the import file. This will be changed to an * when imported. On an export, an * in column 1 for the wildcard of an entire region/jobname will be changed to a ?. When entering the schedule on the panel, use the * in column 1 to specify matching all region/jobnames.

Chapter 9: Utilities

This chapter describes the functionality provided in the Utilities option. Using this option, you can display and update your symbolic files (PROTSYM).

When you select option U of the Primary Option Menu, the Utilities Menu displays. This menu consists of three options - one for listing and displaying the contents of the symbolic file, one for adding listings to a symbolic file, and one for defining programs to be excluded from auto-monitoring.

This section contains the following topics:

Display the Contents of the Symbolic File

After selecting option 1 from the Utilities Menu, the PROTSYM Member Selection panel displays:

```
------------ CA InterTest Batch PROTSYM Member Selection ---------------------
COMMAND ===>                                                  SCROLL ===> CUR
-------------------------------------------------------------------------------
PROTSYM Dsname  ===> 'CAI.PROTSYM'
-------------------------------------------------------------------------------
Cmd  Program   Date       Time       Size Language    Attributes
     ************************** TOP OF DATA ********************************
     CAMRCOBB  12/03/2014 13:56:01     83 COBOL OS/390
     CAMRCOB2  12/03/2014 13:55:47    144 COBOL OS/390
     ITBMAIN   08/16/2014 08:46:02     16 COBOL II
     ITBSUB1   08/06/2014 18:14:59     10 COBOL II
    ************************BOTTOM OF DATA ********************************
```

This panel lists the contents of a given PROTSYM file. From this panel you can enter a valid PROTSYM file in the PROTSYM Dsname field. This displays the contents of this symbolic file. An example is shown in the previous screen. This list displays the program name, the date and time that the program was compiled, the number of records in the PROTSYM that this member is using, the language that was used, and any attributes that the member has.

Using an S line command displays the listing for that particular program.

Add a Program to the PROTSYM File

After selecting option 2 of the Utilities Menu, the PROTSYM Add Program Listings panel displays. Populate a PROTSYM file with one or more listings using this panel.

The PROTSYM Add Program Listings panel contains the following fields:

PROTSYM Dsname

The name of the PROTSYM data set

Listing Dsname

The name of the listing data set

Library Type

The listing library data set type. Specify one of the following:

- PDS–Partitioned Data Set
- SEQ–Sequential Listing File
- PAN–CA-Panvalet Library
- LIB–CA-Librarian Library
- NDV–CA Endevor SCM Library

From Member

The starting member name for the listing data

To Member

The ending member name for the listing data set

View Messages

Identifies the message display options. Specify one of the following:

- ALL–Display all messages
- NONE–Display no messages
- RC–View a single message for each member containing the highest return code

To process just one member, enter the member name in the From Member field. To process a range of members, enter the starting member in the From Member field and the ending member in the To Member. You can also use a prefix ending with an asterisk wildcard to process several members with a common prefix.

You can process a range of members using a batch utility that is described in the member IN25SYMD in the CAMRPROC data set.

Exclude Programs from Auto-monitoring

After selecting option 3 of the Utilities Menu, the Excluded Programs panel displays.

Use this panel to restrict auto-monitoring of programs when wildcarding is used on the Monitor Control panel. Use a single trailing asterisk to define a prefix, if desired. If a program name or wildcarded prefix has been excluded, it can only be monitored by explicitly defining the full name on the Monitor Control panel. To see the list of program names that are already excluded by default, type DEFAULTS on the command line.

Appendix A: Basic Foreground Demo Session

This chapter takes you step-by-step through the basic demo session. Performing the demo at a terminal is the best way to begin learning about this product. If a terminal is not available, you can still learn about the application by reading the chapter as a "paper demo."

This section contains the following topics:

Demo Session Objectives

When you finish the demo session, you know how to:

- Access the application and specify a program you want to test.
- Begin program execution from an Initial Intercept panel.
- Respond to the information provided when the application detects an error.
- Display and dynamically change the value of a data item.
- Resume and control program execution at any point, by setting and removing breakpoints.

Before You Start the Demo

The examples in this guide use default library names and generic references to keystrokes, which differ from installation to installation (for example, the CA Roscoe END key or SPLIT command sequence).

Because the guide cannot anticipate differences at your installation, you need to know certain information specific to your product installation before attempting the examples shown in this manual.

Access the Product

To run the application, you need to know the ISPF option or the CLIST name that calls the application. The examples assume a default ISPF option. Consult your installer to determine the procedure for invoking the application at your installation.

The Demo Source

The source for the demo programs: CAMRPLI, CAMRASM, CAMRCOB, CAMRCOB2, CAMRDMR, CAMRDMR2, and CAMRCOBB are in CAI.CAMRSAMP. Use your compile procedures created during installation to compile a demo program, populate the PROTSYM, and link the program.

CA Roscoe Split Screen Sequence

For the advanced demo session, see the appendix “Advanced Demo Session.” CA Roscoe users need to know the key sequence required to split their one session into two sessions. Consult your CA Roscoe documentation or system programmer.

Begin the Basic Demo

Begin the demo session by accessing the application and selecting the Foreground option from the menu. The sample demo programs are:

CAMRCOB	OS/VS COBOL demo for ISPF users
CAMRCOB2	COBOL II demo for ISPF users
CAMRPLI	PL/I Demo for ISPF users
CAMRASM	Assembler demo for ISPF users
CAMRDMR	OS/VS COBOL demo for CA Roscoe users

CAMRDMR2	COBOL II demo for CA Roscoe users
CAMRCOBB	COBOL II Batch Link Demo

The panels in this chapter illustrate a COBOL II demo session using CAMRCOB2. The screen images of other demos vary from the panels shown in this guide, but the steps for performing the demo are the same.

When testing a program such as CAMRCOB2 or CAMRCOB, the application automatically intercepts all abends. How this works is illustrated later when the application prevents the demo program from abending because of an S0C7.

Access the Primary Option Menu

To access the application, select the option for CA InterTest Batch on your ISPF/PDF Primary Option Menu or enter the name of the CLIST from ISPF Option 6. CA Roscoe users can select the CA InterTest Batch option on the CA Roscoe menu or enter the following commands:

```
[xxx.]IBALLOC
[xxx.]IBRUN
```

where *xxx* is the prefix of the library where the application is installed.

Take a quick look at each of these options before beginning the Demo Session:

Option		Explanation
1	Foreground	Lets you test the execution of an application.
2	Core	Lets you display and alter virtual memory in hexadecimal and character format.
3	Allocation	Lets you allocate the files you use in your test session. This option is not available for CA Roscoe users, who should use the IBCONV RPF for file allocations.
4	Map	Lets you view task control blocks for help in debugging complex system-related problems.
5	Batch	Lets you test an application to be executed in batch.
U	Utilities	Lets you display and update the contents of a PROTSYM.
X	Exit	Terminates the application.

In addition, you can access the tutorial from the Primary Option Menu by pressing your HELP PFkey (normally PF1). The tutorial gives a summary of application facilities and commands.

File Allocation

Each of the demo programs require certain DDs to be allocated before they can be executed correctly. These DDs can be allocated by converting CAI.CAMRJCL(DEMOJCL) to ALIB format and using that ALIB, Use the ALLOC command from within the ALIB editor to allocate those files, or the TSO ALLOC command. The SYSOUT, SYSPRINT, and REPORT are the required DD names and can all be allocated to a SYSOUT class. For more information on allocating DDs to your TSO session or converting JCL to ALIB format, see the chapter "Allocations Facility for ISPF. (see page 171)"

Note: For instructions on converting JCL to CA Roscoe RPF format, see the chapter "JCL Conversion for CA Roscoe (see page 191)."

Specify the Program to be Tested

Begin by selecting the Foreground option.

Action: On the Primary Option Menu, key in: **1**

Press Enter.

Result: The Execution Control Panel displays. This is where you specify the name of the application to be tested.

In the PGM field, enter the name of the program that you are going to debug (CAMRCOB, CAMRCOB2, and so on) In the first Task Libraries field, enter the load library that you linked the demo program into.

Result: The Monitor Control panel displays. The Monitor Control panel lets you specify the name of the programs you are testing and the name of the PROTSYM files where the symbolics for those programs are stored.

Action: Enter the name of the demo program and the PROTSYM file that was used when the program was compiled.

Result: The application accesses the listing for the COBOL program you specified, and displays the *Initial* Intercept panel.

The Initial Intercept Panel

Each time you begin testing, the application displays an *Initial* Intercept (or Breakpoint) panel upon entry to the first program to be tested. The *Initial* breakpoint displays the program listing positioned at the first executable statement. Look at the different parts of this panel.

Note: The text and panels in this guide reference the COBOL II program CAMRCOB2.

The statement line numbers shown in this manual may not exactly match the line numbers you see on your screen.

```
CAMRCOB2 ---------- CA InterTest Batch *INITIAL* Intercept ------------------
COMMAND ===>                                                SCROLL ===> CUR
TRACE=> 000876
      000875        PROCEDURE DIVISION USING PARAMETER-AREA.
      000876        START-PROGRAM.
      000877            OPEN OUTPUT REPORT-OUT.
      000878       * ---------------------------------------------*
      000879       * Welcome to the CA InterTest Batch COBOL II    *
      000880       * Demonstration Program. You are now at the     *
      000881       * Initial Breakpoint panel which is displayed   *
      000882       * upon entry to the first program to be tested. *
      000883       * At this point, other breakpoints can be set   *
      000884       * and control commands can be issued before  the *
      000885       * program is executed.                          *
      000886       * ---------------------------------------------*
      000887            MOVE  R1498-TIMEI TO R1498-TIME.
      000888            PERFORM TIME-EDIT THRU TIME-EDIT-EX.
      000889       *
      000890            MOVE  R1498-TIMEO TO R1498-TIME.
      000891            PERFORM TIME-EDIT THRU TIME-EDIT-EX.
      000892            GO TO DISPLAY-1ST-PANEL.
      000893       * ---------------------------------------------*
      000894       *    THIS ROUTINE EDITS INPUT TIME FOR VALIDITY  *
```

This panel is described next:

The top line of this panel contains 3 parts:

- The left corner displays the name of the program being tested (in this case, CAMRCOB2).
- The middle portion displays the current status. In this case, the status is *INITIAL* since it is the initial entry into the program.
- The right portion is a message display area.

The command Line is where you enter application control commands.

The Trace Line displays up to ten statement numbers. The first number is the statement **about to be executed**, followed by the last nine statements executed. The previous example shows only the statement number about to be executed and the statement number of START-PROGRAM.

The Line Command field is where you enter a single-character command that relates to a specific line of code.

The listing displays and the statement about to be executed is **highlighted**.

Turn the Frequency Display On

The Frequency Counter indicates how many times an instruction was executed in this session. The first time you use the Foreground option, the frequency counter is turned on, but may not be displayed.

To enable the display, enter the following on the command Line:

Action: Key in the command: **freq**

Press Enter.

```
     000881       * Initial Breakpoint panel which is displayed    *
     000882       * upon entry to the first program to be tested.  *
     000883       * At this point, other breakpoints can be set    *
     000884       * and control commands can be issued before  the *
     000885       * program is executed.                           *
     000886       * -----------------------------------------------*
*--> 000887            MOVE  R1498-TIMEI TO R1498-TIME.
*--> 000888            PERFORM TIME-EDIT THRU TIME-EDIT-EX.
     000889       *
*--> 000890            MOVE  R1498-TIMEO TO R1498-TIME.
*--> 000891            PERFORM TIME-EDIT THRU TIME-EDIT-EX.
*--> 000892            GO TO DISPLAY-1ST-PANEL.
     000893       * -----------------------------------------------*
     000894       *    THIS ROUTINE EDITS INPUT TIME FOR VALIDITY  *
```

Result:

A message ? indicates the frequency counter was turned on, and the frequency counter arrows ? are displayed along the left side of the program statements. Since the program has not yet executed any statements, the frequency counter does not display any numbers.

Begin Program Execution

Now you are ready to execute the demo program. The command to execute a program is GO.

Action: Key in the command: **go**

Press Enter.

Result: CAMRCOB2 begins execution and displays the "Welcome Panel" panel.

Action:

Press Enter to continue the demo.

Result:

CAMRCOB2 resumes execution.

Detect and Intercept an Abend

The next screen you see is another type of Intercept panel, an Abend Intercept panel. It tells you that the application halted your program because it detected an error.

```
CAMRCOB2  ---------- CA InterTest Batch ABEND S0C7 Intercept ------------------
COMMAND ===>                                                 SCROLL ===> PAGE
TRACE=> 000938 000935 000933 000931 000925 000920 001291 001290 001283 001282 0
      00937
 0001 00938             ADD +1 TO TASKNUM.
      00939
          ------------------------------------------------------------------
          |        THE FOLLOWING AUTOMATIC BREAKPOINT OCCURRED:            |
          |   INTERCEPT IN PROGRAM CAMRCOB  AT #000938 REASON: ABEND S0C7 |
          |        PRESS ENTER TO REMOVE THIS MESSAGE.                     |
          ------------------------------------------------------------------
 ---> 00945         DISPLAY-2ND-PANEL.
 ---> 00946             CALL 'ISPLINK' USING DSPLY,
      00947                                  SECOND-PANEL.
      00948        * ------------------------------------------- *
      00949        * .. Now to send the third panel which explains  *
      00950        * what we have done so far .. ..                 *
      00951        *                                               *
      00952        * ------------------------------------------- *
 ---> 00953         DISPLAY-3RD-PANEL.
```

Look at the top line of the screen. It tells you the application detected an S0C7 ABEND, and stopped execution because of it. When the application stops program execution, it halts the program at a breakpoint. It can do this automatically, as in the case illustrated, or it can halt a program at a breakpoint that was set by you, the programmer.

Now look at the highlighted ADD instruction. Execution of that ADD instruction triggered the S0C7 ABEND. The application intercepted the abend and then displayed the diagnostic screen you are currently viewing.

The abend type indicates that the problem was caused by improperly formatted data. Which data? Since ADD +1 TO Tasknum triggered the breakpoint, it is likely that Tasknum contains improperly formatted data.

Note: The application intercepted CAMRCOB2 after the abend and identified the problem.

Next:

1. Determine the cause of the error by displaying the current value in Tasknum.
2. Dynamically correct the error by changing the value in Tasknum.

Determine the Cause of the Error

Confirm our guess that the value stored in Tasknum is not valid by displaying its current value.

Action: Tab down to the line containing the highlighted ADD statement. Type **k** to the left of the statement, and press Enter.

```
CAMRCOB2 ---------- CA InterTest Batch ABEND S0C7 Intercept ------------------
COMMAND ===>                                                 SCROLL ===> CUR
TRACE=> 000939 000936 000934 000932 000926 000921 001292 001291 001284 001283 0
      000938
k0001 000939            ADD +1 TO Tasknum.
      000940
      000941       * ---------------------------------------------*
      000942       *                                              *
      000943       * The above statement causes a 0C7.. ..         *
      000944       *                                              *
      000945       * ---------------------------------------------*
 ---> 000946        DISPLAY-2ND-PANEL.
 ---> 000947            CALL 'ISPLINK' USING DSPLY,
      000948                                 SECOND-PANEL.
      000949       * ---------------------------------------------*
      000950       * .. Now to send the third panel which explains  *
      000951       *    what we have done so far .. ..              *
      000952       *                                              *
      000953       * ---------------------------------------------*
 ---> 000954        DISPLAY-3RD-PANEL.
 ---> 000955        CALL 'ISPLINK' USING DSPLY,
```

Result: A Keep Window displays containing the current value of Tasknum, as illustrated in the next panel.

```
CAMRCOB2 ---------- CA InterTest Batch ABEND S0C7 Intercept -----------------
COMMAND ===>                                                 SCROLL ===> CUR
TRACE=> 000939 000936 000934 000932 000926 000921 001292 001291 001284 001283 0
     ------------------------------------------------------------------------
     0E7F24D0 NP-S  000498 03 TASKNUM                    ?00000.
     ------------------------------------------------------------------------
     000938
 0001 000939           ADD +1 TO Tasknum.
     000940
     000941       * ---------------------------------------------*
     000942       *                                              *
     000943       * The above statement causes a 0C7.. ..        *
     000944       *                                              *
     000945       * ---------------------------------------------*
 ---> 000946        DISPLAY-2ND-PANEL.
 ---> 000947           CALL 'ISPLINK' USING DSPLY,
     000948                                SECOND-PANEL.
     000949       * ---------------------------------------------*
     000950       * .. Now to send the third panel which explains  *
     000951       *    what we have done so far....               *
```

If you take a look at the contents of Tasknum, you see that the first character is a question mark. The application displays the question mark when the field does not contain properly formatted data. The value of Tasknum triggered the S0C7 ABEND.

Dynamically Change the Value in Tasknum

Now that you have identified that the value in Tasknum caused the S0C7 ABEND, let us fix it dynamically.

Note: In real testing, you might search the program listing or program execution path to investigate the source of the error, and perhaps even update your program code using a split-screen. These tasks are all performed in the advanced demo session in the appendix “Advanced Demo Session.” For now, assume that zero was the value intended, correct the value of Tasknum, and continue our testing.

Action: Key in the command: **set tasknum = 0**

Press Enter**.**

Result: The application dynamically sets the value of Tasknum to zero, as indicated in the message that displays in the top-right corner of the screen: **SET COMPLETE**. You can see the new value of Tasknum in the Keep Window (+00000.).

```
CAMRCOB2 ---------- CA InterTest Batch ABEND S0C7 Intercept ------ SET COMPLETE
 COMMAND ===>                                                  SCROLL ===> CUR
 TRACE=> 000939 000936 000934 000932 000926 000921 001292 001291 001284 001283
      ------------------------------------------------------------------------
      0E7F24D0 NP-S  000498 03 TASKNUM                      +00000.
      ------------------------------------------------------------------------
      000938
  0001 000939           ADD +1 TO Tasknum.
      000940
      000941       * ---------------------------------------------*
      000942       *                                              *
      000943       * The above statement causes a 0C7.. ..        *
      000944       *                                              *
      000945       * ---------------------------------------------*
  ---> 000946        DISPLAY-2ND-PANEL.
  ---> 000947           CALL 'ISPLINK' USING DSPLY,
      000948                                SECOND-PANEL.
      000949       * ---------------------------------------------*
      000950       * .. Now to send the third panel which explains  *
      000951       *    what we have done so far .. ..            *
      000952       *                                              *
      000953       * ---------------------------------------------*
  ---> 000954        DISPLAY-3RD-PANEL.
  ---> 000955           CALL 'ISPLINK' USING DSPLY,
      000956                                THIRD-PANEL.
      000957       *
```

Note: You do not have to know the type of data (binary, packed, and so on) or the length of Tasknum to correctly set its value. The SET command is comparable to a MOVE statement that automatically takes care of different data types and their lengths.

Control Program Execution

Now that the value in Tasknum has been properly initialized to zero, you could continue testing by resuming program execution.

This is also a good opportunity to learn about an important feature: your ability to control program execution by setting breakpoints.

How to Stop Your Program by Setting Breakpoints

One of the problems with traditional testing methods is that you have little or no control over program processing. You simply initiate the program, and the program either runs to completion, loops, or abends.

With this product, you can control program execution in a number of ways. For example, you can set "stops," called breakpoints, anywhere in your program. You can set two types of breakpoints: unconditional and conditional.

- When you set an unconditional breakpoint at a statement, the program stops just before the statement is executed.
- When you set a conditional breakpoint at a statement, the program stops only if the value of a data item changes or a condition you specified is met, such as a data-item equaling or exceeding some value.

What You Can Do When Your Program Is Stopped

Once a program is stopped, you can use the application's testing and debugging facilities to do the following:

- Examine the listing.
- Examine and modify main storage to detect and correct errors.
- Set and remove breakpoints.
- “Go around a problem” by skipping one or more statements, or by resuming execution from a location other than the one at which the program is currently stopped.
- Execute the program in single-step mode; that is, the program executes only one verb (or the number of verbs specified in the step count), and then stops again.
- Terminate program execution.

Note: All of these activities are described in detail in the advanced demo session in the appendix “Advanced Demo Session.”

The next section demonstrates how easy it is to control program execution by setting an unconditional breakpoint.

Set Unconditional Breakpoints

You can set unconditional breakpoints directly on any breakpoint display using the line command u, or the command UNCOND.

See how to set a breakpoint before the execution of the DISPLAY-2ND-PANEL using the **u** line command.

Action: Type **u** to the left of the paragraph name DISPLAY-2ND-PANEL.

Type **GO** to continue execution from where the program is currently stopped (at the statement ADD +1 TO Tasknum).

```
CAMRCOB2 ---------- CA InterTest Batch ABEND S0C7 Intercept ------ SET COMPLETE
COMMAND ===>                                                  SCROLL ===> CUR
TRACE=> 000939 000936 000934 000932 000926 000921 001292 001291 001284 001283 0
     ------------------------------------------------------------------------
     0E7F24D0 NP-S  000498 03 TASKNUM                    +00000.
     ------------------------------------------------------------------------
     000938
 0001 000939            ADD +1 TO Tasknum.
     000940
     000941       * ---------------------------------------------*
     000942       *                                              *
     000943       * The above statement causes a 0C7.. ..        *
     000944       *                                              *
     000945       * ---------------------------------------------*
u---> 000946        DISPLAY-2ND-PANEL.
 ---> 000947            CALL 'ISPLINK' USING DSPLY,
     000948                                 SECOND-PANEL.
     000949       * ---------------------------------------------*
     000950       * .. Now to send the third panel which explains  *
     000951       *    what we have done so far .. ..            *
     000952       *                                              *
     000953       * ---------------------------------------------*
 ---> 000954        DISPLAY-3RD-PANEL.
```

Result: The application resumes program execution and continues until it reaches the breakpoint that you just set.

The application halts the program before the first executable statement in the paragraph is executed and displays the screen.

```
CAMRCOB2 ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
COMMAND ===>                                                 SCROLL ===> PAGE
TRACE=> 000949 000948 000941 000938 000936 000934 000928 000923 001294 001293 0
     ------------------------------------------------------------------------
     001507E0 NP-S  000500 02 TASKNUM                    +00001.
     ------------------------------------------------------------------------
       000940
 0001  000941                   ADD +1 TO Tasknum.
       000942
       000943             * -----------------------------------------------*
       000944             *                                                *
       000945             * The above statement causes a 0C7.. ..          *
       000946             *                                                *
       000947             * -----------------------------------------------*
```

```
U--->  000948              DISPLAY-2ND-PANEL.
 --->  000949                  CALL 'ISPLINK' USING DSPLY,
       000950                                       SECOND-PANEL.
       000951             * -----------------------------------------------*
       000952             * .. Now to send the third panel which explains  *
       000953             *    what we have done so far .. ..              *
       000954             *                                                *
       000955             * -----------------------------------------------*
 --->  000956              DISPLAY-3RD-PANEL.
 --->  000957                  CALL 'ISPLINK' USING DSPLY,
       000958                                       THIRD-PANEL.
```

Note that the top line of the screen now identifies the reason for stopping as UNCOND BEFORE INTERCEPT. This indicates that the application has halted the program at a breakpoint you set.

Notice the U to the left of the paragraph DISPLAY-2ND-PANEL. It identifies the breakpoint as an Unconditional breakpoint. The first executable statement in that paragraph is the CALL statement. This is the statement that is about to be executed, or the *current statement*.

When you are stopped at a breakpoint, you can do the following:

- Scroll or search through your source listing.
- Examine and modify storage.
- Set and remove breakpoints using line commands such as **u** and **x**.
- Go around a problem by resuming program execution from another location.
- Terminate program execution and stop the test session.

When debugging your own programs, you typically perform one or more of these activities and they are described in detail in the next chapter. However, CAMRCOB2 does not have any more errors, so we are just going to complete the demo session.

Remove the Breakpoint and Keep Window

As you continue testing and debugging, it is a good practice to remove breakpoints you no longer need, so that the program will not stop unnecessarily. Remove the unconditional breakpoint you just set so CAMRCOB2 will not stop before each execution of DISPLAY-2ND-PANEL in our test session. To reset the display, we are also going to remove the Keep Window.

Action: Key in the following command to remove the Keep Window: **remove all**

Tab down and type over the U with an **x** to remove the breakpoint.

Press Enter.

```
CAMRCOB2 ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
COMMAND ===>                                                 SCROLL ===> CUR
TRACE=> 000946 000939 000936 000934 000932 000926 000921 001292 001291 001284 0
      ------------------------------------------------------------------------
      0E7F24D0 NP-S  000498 03 TASKNUM                     +00001.
      ------------------------------------------------------------------------
      000938
 0001 000939            ADD +1 TO Tasknum.
      000940
      000941       * ---------------------------------------------*
      000942       *                                              *
      000943       * The above statement causes a 0C7.. ..        *
      000944       *                                              *
      000945       * ---------------------------------------------*
x---> 000946        DISPLAY-2ND-PANEL.
 ---> 000947            CALL 'ISPLINK' USING DSPLY,
      000948                                 SECOND-PANEL.
      000949       * ---------------------------------------------*
      000950       * .. Now to send the third panel which explains  *
      000951       *    what we have done so far .. ..            *
      000952       *                                              *
      000953       * ---------------------------------------------*
 ---> 000954        DISPLAY-3RD-PANEL.
```

Result: The application removes the Keep Window and the unconditional breakpoint.

Notes on the **remove** command:

- If you have more than one data item displayed in the Keep Window, you can specify a single item to be removed. For example, to remove just the line that displays Tasknum, you enter the command: **remove tasknum**.
- The **r** line command also removes all items in the Keep Window that are contained in the statement to the right of the line command. For example, you tab down to the statement containing Tasknum, type **r**, and press Enter.

Continue Execution

Action: Type **GO** to continue program execution.

Result: CAMRCOB2 next displays a screen confirming that you have successfully corrected the problem in the demo program.

Action Press Enter.

Result: CAMRCOB2 displays the following demo summary screen.

Action: To complete this part of the sample test session, press Enter.

Result: CAMRCOB2 displays the demo exit screen.

Action: Exit the demo program by pressing Enter.

Result: The Execution Control Panel displays. Each time you complete or quit a foreground test session, the application returns you to the Execution Control Panel with the message:

- ENDED, RETURN CODE=XX

Note: To end the demo session, exit the application using the CA Roscoe END-key (usually PF3) or the ISPF END-key or command.

To continue with the advanced demo session, skip to the appendix “Advanced Demo Session”.

What You Have Learned

The demo session has taught you the basics of using this product to test a program. You have learned how to:

- Specify the program you want to test.
- Begin program execution.
- Interpret the diagnostic information the application provides when it detects an error.
- Display and modify main storage.
- Set and remove a breakpoint.
- Resume program execution.

Appendix B: Basic Batch Link Demo

This chapter takes you step-by-step through the basic Link demo session. Performing the demo at a terminal is the best way to begin learning about the application. If a terminal is not available, you can still learn about this product by reading the chapter as a "paper demo."

The basic demo session illustrates many of the testing and debugging tasks you use on your own programs. For more information on these tasks, see Testing Your Programs Using Batch Link.

This section contains the following topics:

Demo Session Objectives

This demo session builds on what you have learned in the last chapter, "Basic Foreground Demo Session." The focus in this chapter is on using the Batch Link features:

- Select JCL to execute the program you want to debug.
- Select a running job that is waiting to be debugged.
- Connect your terminal to a batch job and begin debugging that program.

Before You Begin

The source for the Batch Link Demo program, CAMRCOBB, is in CAI.CAMRSAMP. Use the compile procedure you customized during the install to compile, link, and post-process the COBOL program CAMRCOBB.

Begin the Batch Link Demo

Begin the demo session by accessing the application and selecting the Batch Link Facility option from the menu. This brings you to the Batch Link Option Menu.

From this point, you can select the JCL that executes the program that you want to run in batch. Select Option 1. The Batch Link JCL Conversion panel displays.

On this panel, you can provide the JCL member that you want to execute in batch. For the demo, enter CAI.CAMRJCL(DEMOJCL) in the Other Partitioned Data Set field. Note that your installer may have changed CAI to another high-level qualifier; make sure that the data set name is correct.

After entering the JCL that you want to submit, the Batch Link Conversion Options panel displays.

Here you can specify any of the conversion options. For a full description of these options, see the chapter "Batch Link Facility." For the purposes of the demo, no changes are required on this panel. Press Enter. If this were a multi-step job, you would see a STEP SELECTION panel where you could select one or more steps to be debugged. This JCL stream, however, contains just one step.

The next panel shows the converted JCL:

```
//DEMOJCL  JOB  (UNKNOWN),UNKNOWN  *** JOB STATEMENT SUPPLIED ***
//INSTRUCT DD   *
*--------------------------------------------------------------------
*    BATCH LINK DEMO INSTRUCTIONS                                   *
*                                                                   *
*    AFTER MODIFYING THIS JCL TO CONFORM TO YOUR INSTALLATION       *
*    STANDARDS, PLEASE DELETE THIS INSTRUCT DD STATEMENT.           *
*                                                                   *
*    1. MODIFY THE JOB CARD FOR YOUR INSTALLATION.                  *
*                                                                   *
*    2. MODIFY THE HIGH LEVEL QUALIFIER FOR THE IVP.LOAD            *
*       LIBRARY IN STEPLIB.                                         *
*                                                                   *
*-------------------------------------------------------------------*
//STEP1    EXEC PGM=CAMRBL01,REGION=4M
//INT1OPTS DD *
  EXEC=CAMRCOBB,PROFILE=USER01
/*
//INT1PARM DD DISP=SHR,DSN=CAI.CAMRSAMP
//INT1LOAD DD DISP=SHR,DSN=CAI.CAMRLOAD
//INT1PNLL DD DISP=SHR,DSN=CAI.CAMRPNL1
//INT1MSGL DD DISP=SHR,DSN=CAI.CAMRMSG0
//INT1PROF DD DISP=SHR,DSN=CAI.PROFLIB
//INT1CLIB DD DISP=SHR,DSN=CAI.CAMRSRC
//INT1CLOG DD SYSOUT=*
//INT1REPT DD SYSOUT=*
//STEPLIB  DD DISP=SHR,DSN=CAI.CAMRLOAD
//         DD   DISP=SHR,DSN=CAI.IVP.LOAD    <=== CHANGE
//SYSPRINT DD   SYSOUT=*
//SYSOUT   DD   SYSOUT=*
```

Make the necessary changes to the JCL as documented in the JCL. This includes adding a job card to the JCL, changing the “CAI” high level qualifier of the IVP load library to the correct high level qualifier and removing the INSTRUCT DD. When these changes have been made, submit the JCL. You return to the Batch Link Option menu. The demo program, CAMRCOBB, is now waiting to be debugged. Select option 2 to link your ID to the batch job. If, after selecting option 2, you get a message NO BATCH JOBS AVAILABLE, the job you just submitted has not started executing yet.

When it does execute, you see a screen that looks similar to the following panel:

```
----------------- CA InterTest Batch Batch Link Selection --------------------
COMMAND INPUT ===>                                          SCROLL ===> PAGE
          *         *         *         *         *         *         *
     #     Jobname   Stepname  Program   Owner     User      Trans     SYSID
     ************************** TOP OF DATA ********************************
     1     JOB01A    STEP1     TESTPGM   USER01                        TSO01
     2     JOB02A    STEP1     CAMRCOBB  USER02                        TSO02
     3     JOB03X    RUN       PAYROLL   USER03   LINKED BY USER03     TSO01
     ************************* BOTTOM OF DATA ******************************
```

This screen lists programs that are either currently being debugged or waiting to be linked to a terminal. In this case, there are three programs executing, two waiting to be debugged. Select the job that you submitted in batch by typing an **S** next to it and pressing Enter. The Monitor Control panel displays.

Enter CAMRCOBB in the monitored programs list and enter the PROTSYM that contains the symbolics for CAMRCOBB in the Symbolic Files list. As with the foreground demo, after pressing Enter, the Initial Intercept panel displays. From this point, the use of the Batch Link feature does not differ from using the product in foreground. You can set breakpoints, change the value of data items, control program execution, split your screen, and so on.

Type **GO** to begin execution of the program. You stop at COBOL statement ADD 1 TO TASKNUM due to an S0C7 ABEND Intercept. Set the value of TASKNUM to a valid value using the SET command: SET TASKNUM = 1

Type **GO** and the program runs to completion.

You return to the Batch Link Option panel. The completion code of the step is indicated in the upper right corner of the screen.

Test Your Programs Using Batch Link

Note: You must first populate a PROYSYM to create the symbolics for any program you want to debug. This can be done by compiling and post-processing the program using the procs customized during the install or by using the Utility option 2, ADD. For more information on the Utility option, see the chapter "Utilities."

Access CA InterTest Batch and select the Batch Link Facility option from the menu. The Batch Link Option Menu displays.

Select Option 1, which lets you prepare existing JCL to be submitted in batch for debugging. The Batch Link JCL Conversion panel displays.

Specify where the JCL is located. If your JCL is stored in a CA Panvalet or a CA Librarian data set, indicate that on this panel. Press Enter. The Batch Link Conversion Options panel displays.

Enter any system PROCs that are required for this JCL to execute or provide a pre-allocated DD that contains these PROCs. For a description of the fields on this panel, see the chapter “Batch Link Facility.” Press Enter.

The Step Selection List panel displays:

```
--------------------- CA InterTest/Batch Step Selection List -------------------
COMMAND INPUT ===>                                            SCROLL ===> PAGE
     Select one or more of the following Steps:
     ***************************** TOP OF DATA ********************************
     1 //STEP1 EXEC PGM=ISFBR14
     2 //STEP2 EXEC PGM=TEST2,REGION=4M
   S 3 //RUN   EXEC PGM=COB0C7,REGION=4M,PARM='/RPTOPTS(ON)'
     ************************** BOTTOM OF DATA ******************************
```

The Step Selection List panel displays a list of steps from the JCL you selected to execute. Select the steps that you want to debug. In the previous example, there are three steps: a compile, a link, and execute. Select the execute step. Press Enter. The converted JCL is displayed.

Review the JCL to make sure it is the job that you want to submit in batch. Type **SUB** to submit the job. Pressing PF3 will take you to the Batch Link Option Menu.

Select option 2. This displays a list of programs that are either currently being debugged or waiting to be linked to a terminal. Select the job that you submitted in batch by typing **S** next to it and pressing Enter. The Monitor Control panel displays.

Enter your program in the Monitored Programs list and your PROTSYM in the Symbolic Files list. Press Enter to debug your program.

What You Have Learned

This Batch Link Demo has taught you the basics of using this product to test a program running in batch. You have learned how to:

- Submit converted JCL in batch.
- Link a program running in batch to your terminal.
- Debug a program running in batch.
- Test a program running in batch.

Appendix C: Advanced Demo Session

While the appendices "Basic Foreground Demo Session" and "Basic Batch Link Demo" of this guide illustrate the basics of working with the application, the sessions presented here let you use some of the more advanced features. This part of the demo is also modular; you choose the option you wish to perform from a menu. Each option illustrates a different feature, summarized as follows:

Advanced Demo Session Options	
Option 1	Perform time-driven execution.
Option 2	Set conditional breakpoints.
Option 3	Update source code.
Option 4	Interrupt a program loop.
Option 5	Trace program execution.
Option 6	Work with indexed tables. **Note:** This option is not available in the Assembler demo.
Option 7	Generate a histogram report.

Notes:

Before accessing the Demo Session Options Menu, you must complete the setup steps outlined in the next section, Advanced Demo Preliminaries.

You also need to know if the names of the following libraries have been changed at your installation:

- CAMRPROC
- CAMRSRC
- ALIB
- INCLIB

This section contains the following topics:

Advanced Demo Preliminaries

The advanced demo session uses the same program used in the basic demo session: CAMRASM for Assembler users, CAMRCOB2 or CAMRDMR2 for COBOL II and higher users, CAMRCOB or CAMRDMR for COBOL/VS users, and CAMRPLI for PL/I users. To begin the advanced demo session, you need to do the following setup steps:

- Allocate data sets to your test session.
- Set a number of breakpoints from the Initial Intercept panel.

After executing the program, you branch to the Demo Session Options Menu. The setup steps are described next.

Access the *Initial* Intercept Panel for the Demo Program

If you just completed the basic demo, the application brings you to the Execution Control panel, with your previous entries for the Basic Demo displayed. From this point, do the following actions

Action: Press Enter.

Result: The Monitor Control Entry panel displays with your previous entries.

Action: Press Enter.

Result: The Initial Intercept panel displays. Continue with the section, Split Your Screen.

If you have not just completed the basic demo, follow the beginning section of the appendix “Basic Foreground Demo Session” to get to the Initial Intercept panel. The sections to follow include:

- Beginning the basic demo
- Accessing the Primary Option Menu
- File allocation
- Specifying the program to be tested

Continue with the next section, Split Your Screen.

Split Your Screen

Action: **ISPF users:**

Tab a few lines down your screen and press the SPLIT PF key defined in your ISPF profile (normally PF2).

CA Roscoe users:

Enter the CA Roscoe SPLIT sequence.

Result: **ISPF users:**

The ISPF Primary Option Menu displays in the bottom half of your screen.

CA Roscoe users:

A second CA Roscoe panel displays.

The CAMRCOB2 Initial Intercept Panel with split-screen (ISPF users) is shown next.

```
CAMRCOB2 ---------- CA InterTest Batch *INITIAL* Intercept -------------------
COMMAND ===>                                                 SCROLL ===> CUR
TRACE=> 000876
      000875        PROCEDURE DIVISION USING PARAMETER-AREA.
 *--> 000876        START-PROGRAM.
 *--> 000877            OPEN OUTPUT REPORT-OUT.
      000878       * ---------------------------------------------*
      000879       * Welcome to the CA InterTest Batch COBOL II    *
      000880       * Demonstration Program. You are now at the     *
      000881       * Initial Breakpoint panel which is displayed   *
      000882       * upon entry to the first program to be tested. *
      000883       * At this point, other breakpoints can be set
*       .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
-----  -------  ------  ISPF/PDF PRIMARY OPTION MENU
-----------------------       VERSION 3 RELEASE 1         ------------------
OPTION  ===>
                                                          USERID   -xxxxxxx
   0  ISPF PARMS  - Specify terminal and user parameters   TIME     -14:37
   1  BROWSE      - Display source data or output listings  TERMINAL -3278
   2  EDIT        - Create or change source data            PF KEYS  -24
   3  UTILITIES   - Perform utility functions              DATE(G)  -14/12/18
   4  FOREGROUND  - Invoke lang. processors in foreground  DATE(J)  -14.352
   5  BATCH       - Submit job for language processing
   6  COMMAND     - Enter TSO command or CLIST
   7  DIALOG TEST - Perform dialog testing
```

Advanced Demo Data Sets

The advanced demo requires the following two additional data sets:

- User report data set (ddname: INT1REPT)
- Command library data set (ddname: INT1CLIB)

Use the user report data set to store the HISTOGRAM and XSUM reports, which are used in Option 7 of the advanced demo.

Create and Allocate INT1REPT Data Sets

Users must define their own INT1REPT data set. Create the INT1REPT data set using one of the following methods.

Method 1

1. Use ISPF or CA Roscoe to enter the JCL shown in the sample INT1REPT Create in the following example:

```
//yourjob   JOB
//ALLOC     EXEC PGM=IEFBR14
//INT1REPT  DD DSN=userid.INT1REPT,DISP=(,CATLG),
//          DCB=(RECFM=FBA,LRECL=131,BLKSIZE=6157),
//          UNIT=SYSDA,
//          SPACE=(TRK,(30,5))
```

2. Submit the job to create the data set.
3. Check the condition code of the job to verify that the INT1REPT data set was created.
4. Perform one of the following commands to allocate the data set:
   - **(**ISPF users)
     Select Option 6 (command) from the ISPF Primary Menu and key in the command:

     ALLOC DD(INT1REPT)DATASET('userid.INT1REPT')SHR
   - (CA Roscoe users)

     ALLOC INT1REPT DSN=userid.INT1REPT DISP=SHR
5. Press Enter.

   TSO or CA Roscoe allocates the data set.

Method 2

1. Use option 3.2 from the ISPF/PDF Primary Option Menu to create the INT1REPT data set.

   Use the information in the Method 1 JCL when allocating INT1REPT. In the Method 1 JCL, in the SPACE definition, 30 is the primary space and 5 is the secondary.

   For additional help using option 3.2, ISPF provides excellent online help that is accessible with the PF1 key.

2. Check to verify that the INT1REPT data set was created.

3. Perform one of the following commands to allocate the data set:

   - **(**ISPF users)

     Select Option 6 (command) from the ISPF Primary Menu and key in the command:

     ```
     ALLOC DD(INT1REPT)DATASET('userid.INT1REPT')SHR
     ```

   - (CA Roscoe users)

     ```
     ALLOC INT1REPT DSN=userid.INT1REPT DISP=SHR
     ```

4. Press Enter.

   TSO or CA Roscoe allocates the data set.

Method 3

- Select Option 6 (command) from the ISPF Primary Menu and allocate INT1REPT to a SYSOUT class with the following command:

  ```
  ALLOCATE FI(INT1REPT) SYSOUT(x)
  ```

  x is the sysout class you want the reports to be written to.

  INT1REPT is allocated to SYSOUT.

Note: A CLIST, INT1REPT, exists that creates a user report data set.

INT1CLIB

The command library data set, which was created during the installation, can be shared by all users. The default name for this data set is CAI.CAMRSRC, but this may have been changed by your installer. INT1CLIB should be allocated automatically. Consult your installer to verify that this is the case.

Three of the members in the command library contain include commands to set the breakpoints required for the advanced demos. They are:

- ASMINCL commands for Assembler demo
- DEMOINCL commands for COBOL demos
- PLIINCL commands for PL/I demo

Allocate INT1CLIB Data Sets

INT1CLIB data sets should be allocated automatically. In case this does not happen, these instructions describe how to allocate command library data sets.

To allocate the INT1CLIB data set

1. Key in one of the following commands:
   - (ISPF users)

     Select Option 6 (command) from the ISPF Primary Menu and key in the command:

     ```
     ALLOC DD(INT1CLIB) DATASET('CAI.CAMRSRC')SHR
     ```

   - (CA Roscoe users)

     ```
     ALLOC INT1CLIB DSN=CAI.CAMRSRC DISP=SHR
     ```

2. Press Enter.

   TSO or CA Roscoe allocates the data set.

   Note: Your system installer may have changed CAI to another high-level qualifier. Check with your installer.

Advanced Demo Session for COBOL

This section illustrates a COBOL advanced demo session using CAMRCOB2. The screen images for the other COBOL demos may vary from those shown here, but the steps for performing the demo are the same.

Include Unconditional Breakpoints

Now you are going to set unconditional breakpoints at a number of procedure names in the demo program. In the basic demo you set an unconditional breakpoint using the **u** line command, which is equivalent to the **UNCOND** command. However, a quicker way to include unconditional breakpoints for a test session is to place the breakpoint commands in a data set member, and simply include that member when the Initial Intercept panel displays.

The DEMOINCL data set member of CAI.CAMRSAMP contains the application commands that set the breakpoints required for using the Advanced Options Demo for COBOL. Include this data set member as follows:

Action: Enter the following command at the *Initial* Intercept panel:

include demoincl

Press Enter.

Result: The application processes the commands in the demoincl data set, and returns the following message in the upper-right corner of the screen: BREAKPOINT SET.

```
CAMRCOB2 ---------- CA InterTest Batch *INITIAL* Intercept ----- BREAKPOINT SET
COMMAND ===>                                                  SCROLL ===> CUR
TRACE=> 000876
      000875        PROCEDURE DIVISION USING PARAMETER-AREA.
 *--> 000876        START-PROGRAM.
 *--> 000877            OPEN OUTPUT REPORT-OUT.
      000878       * ---------------------------------------------*
      000879       * Welcome to the CA InterTest Batch COBOL II    *
      000880       * Demonstration Program. You are now at the     *
      000881       * Initial Breakpoint panel which is displayed   *
      000882       * upon entry to the first program to be tested. *
      000883       * At this point, other breakpoints can be set   *
      000884       * and control commands can be issued before  the *
      000885       * program is executed.                          *
      000886       * ---------------------------------------------*
 *--> 000887            MOVE  R1498-TIMEI TO R1498-TIME.
 *--> 000888            PERFORM TIME-EDIT THRU TIME-EDIT-EX.
      000889       *
 *--> 000890            MOVE  R1498-TIMEO TO R1498-TIME.
 *--> 000891            PERFORM TIME-EDIT THRU TIME-EDIT-EX.
 *--> 000892            GO TO DISPLAY-1ST-PANEL.
      000893       * ---------------------------------------------*
      000894       *    THIS ROUTINE EDITS INPUT TIME FOR VALIDITY  *
```

Access the Demo Session Options Menu

After setting the required breakpoints, you can access the Demo Session Options Menu by executing the demo program and choosing the Advanced Options Menu from the Welcome panel. Use the following steps:

Action: From the Initial Intercept panel, type **GO** and press Enter to execute the program.

Result: The demo program begins execution and displays the Welcome panel.

Action: Press **PF2** to access the Options Menu for the Advanced Demo Session.

Result: The Advanced Options Preliminary Screen displays, reminding you not to access the Options Menu without including the data set member DEMOINCL.

Action: Press Enter to continue.

Result: The Demo Session Options Menu displays.

Each of the Options on the menu is independent and can be performed in any order. To use an Option, follow the appropriate section. Upon completing an Option, you return to the Menu.

Before you start you should know that to interrupt a loop you must press the ATTN or PA1 key. Try the ATTN key first. If that does not interrupt the loop, your testing environment requires you to use the PA1 key. If using PA1, you may need to press RESET first.

Exit the Advanced Options Demo

You can exit the demo from the Options Menu at any time.

Action: Press **PF3**.

Result: The End Demo Session screen displays.

Action: Press Enter.

Result: The Program Exit Intercept panel displays.

Action: Type **GO** and press Enter to execute the program.

Result: The demo program ends, and the Entry panel displays.

Return to the Advanced Options Demo Session

If you exit the demo, you can return at any time to perform another Advanced Option. Just repeat the preliminary steps given in the beginning of this chapter. Remember to enter the following command at the Initial Breakpoint screen:

```
include demoincl
```

Option 1: Time-Controlled Execution

This option illustrates how you can slow the execution of your program. The SLOW command executes one verb in your program every few seconds. The exact time interval is specified when you issue the SLOW command. Using the SLOW command together with a keep window is an especially effective testing technique; it allows you to halt the program whenever you see a problem, without having to determine the breakpoint in advance.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

This option requires you to interrupt the program by pressing the ATTENTION key or the PA1 key. Before proceeding, locate each of these keys on your keyboard. If you need help in finding them, ask a technical support person **before** you begin.

Action: Key in Option **1**.

Press Enter.

Result: CAMRCOB2 displays the panel that describes what occurs in this part of the demo session.

Action: Press Enter.

Result: The application halts CAMRCOB2 at an unconditional breakpoint set at the paragraph PAY-CALC. The first executable statement in that paragraph is the current statement, and is highlighted.

Open a keep window that displays the values of YTD and X-AXIS by entering commands shown following. Notice how you can use the short form of the keep command (k) and use the TSO command delimiter (normally a semi-colon) to string multiple commands on a single command line.

Action: Key in the following command:

k ytd; k x-axis

Result: CA InterTest Batch displays a keep window showing the current values of YTD and X-AXIS.

A sample screen showing the keep window for YTD and X-AXIS follows:

```
CAMRCOB2 ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
COMMAND ===>                                                 SCROLL ===> CUR
TRACE=> 001055 001053 001052 001051 001049 001043 001042 001040 001035 001017 0
     ------------------------------------------------------------------------
     1F08CB11 ND-OT 000683 03 YTD                          +0000000.00
     1F08C8E0 AN    000553 05 X-AXIS
     ------------------------------------------------------------------------
     001054       *
U---> 001055        PAY-CALC.
 ---> 001056            PERFORM POPULATE-GRAPH THRU POPULATE-GRAPH-EX
     001057                 VARYING SUB-4 FROM 1 BY 1
     001058                     UNTIL SUB-4 GREATER THAN NO-OF-HOURS.
 ---> 001059        PAY-CALC-EX.
 ---> 001060            MOVE SPACES      TO CHEQUE-LINE.
 ---> 001061            MOVE TOTAL-GROSS TO CHEQUE-AMOUNT.
 ---> 001062        PAY-RETURN.
 ---> 001063            GO TO OPTIONS.
     001064       *
 ---> 001065        POPULATE-GRAPH.
 ---> 001066            MOVE MONTH-ITEM(SUB-4)     TO X-AXIS-YTD.
 ---> 001067            MOVE TOKEN-ITEM            TO X-AXIS-R(SUB-4).
 ---> 001068            ADD  MONTHLY-AMOUNT(SUB-4) TO YTD.
 ---> 001069            MOVE YTD                   TO TOTAL-GROSS.
```

Issue the SLOW command to execute one verb every two seconds. As the program resumes execution, you are able to see the changing values of YTD and X-AXIS. In this demo, you are going to halt execution when the X-AXIS value is MAR by pressing the ATTENTION key.

Action: Key in the command: **slow 2**

Press Enter.

Result: The application initiates a time-controlled execution of CAMRCOB2, at a rate of one verb every two seconds.

Action: When the value of X-AXIS in the keep window is **MAR**, press the **ATTN** or **PA1** key. Try the **ATTN** key first.

Result: The Attention Intercept panel displays, as shown in the following panel.

A sample Attention Intercept panel follows:

```
CAMRCOB2 ---------- CA InterTest Batch ATTENTION Intercept ------------------
COMMAND ===>                                                 SCROLL ===> CUR
TRACE=> 001069 001068 001067 001066 001065 001071 001070 001069 001068 001067 0
     ------------------------------------------------------------------------
     1F08CB11 ND-OT 000683 03 YTD                          +0255273.52
     1F08C8E0 AN    000553 05 X-AXIS                       Mar ->->->
     ------------------------------------------------------------------------
     001054       *
U0001 001055        PAY-CALC.
 0001 001056            PERFORM POPULATE-GRAPH THRU POPULATE-GRAPH-EX
     001057                 VARYING SUB-4 FROM 1 BY 1
     001058                     UNTIL SUB-4 GREATER THAN NO-OF-HOURS.
 ---> 001059        PAY-CALC-EX.
 ---> 001060            MOVE SPACES      TO CHEQUE-LINE.
 ---> 001061            MOVE TOTAL-GROSS TO CHEQUE-AMOUNT.
 ---> 001062        PAY-RETURN.
 ---> 001063            GO TO OPTIONS.
     001064       *
 0003 001065        POPULATE-GRAPH.
 0003 001066            MOVE MONTH-ITEM(SUB-4)     TO X-AXIS-YTD.
 0003 001067            MOVE TOKEN-ITEM            TO X-AXIS-R(SUB-4).
 0003 001068            ADD  MONTHLY-AMOUNT(SUB-4) TO YTD.
 0002 001069            MOVE YTD                   TO TOTAL-GROSS.
 0002 001070        POPULATE-GRAPH-EX.
```

Review what you just did.

- You opened a keep window for two variables whose values you wished to view and compare.
- You used the command **slow 2** to have the application slowly execute the program. However, during the program execution, you could monitor the values of the variables in the keep window.
- You used the ATTENTION or PA1 key to stop program execution, and the Attention Intercept panel displays.

This combination of using keep windows, the slow command, and an Attention Intercept allows you to carefully control and monitor your program execution, while tracking the values of specific variables.

Note: The Attention Intercept panel is only displayed when you press the ATTN or PA1 key and then the application executes a statement in a program that is currently being monitored.

Remove the Keep Window

Before returning to the Options Menu of the advanced demo, remove the keep window as follows:

Action: Key in the command: **remove all**

Press Enter.

Result: The application removes the keep window for YTD and X-AXIS on the Attention Intercept panel.

Continue Execution

Action: Type **GO** and press Enter to continue execution.

Result: CAMRCOB2 resumes execution, and displays the Options Menu.

This concludes Option 1: Time Controlled Execution. Select another option and go to the appropriate section in this guide, or exit the demo using **PF3**.

Option 2: Set Conditional Breakpoints

This option shows you how to set and use conditional breakpoints.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

Action: Key in Option **2**.

Press Enter.

Result: CAMRCOB2 displays the screen that describes what occurs in this part of the demo session.

Action: Press Enter.

Result: The application halts CAMRCOB2 at an unconditional breakpoint. The first executable statement in the paragraph is highlighted.

Access the screen used to set **conditional** breakpoints. From this screen, you can set both the conditions and any commands you want executed when the specified condition is met.

Action: Key the following in the Command field: **cond**

Press Enter.

Result: The application displays the When panel. WHEN is a synonym for COND.

Action: Make the following entries on the When panel:

Press Enter.

```
----------------------- CA InterTest Batch WHEN PANEL ------------------------
OPTION ===>S

   S - SET A WHEN CONDITION
   R - RESET A WHEN CONDITION
   L - OR BLANK, LIST THE WHEN CONDITIONS

SPECIFY DATA AREA NAME(S) AND COMPARISON CONDITION:
   WHEN-NAME    ===> ytdcond
   DATA-AREA-1  ===> sub-total
   OPERATOR     ===> ge       (EQ, NE, LT, GT, LE, GE, =, <>, <, >, <=, >=)
   DATA-AREA-2  ===> 600.00
   AFTER        ===> N         (Y/N)

COMMANDS TO BE EXECUTED AT WHEN CONDITION:
===>  k sub-total; set total-gross=2000; k total-gross
```

Result: The application displays the following message in the top right corner of the When panel: WHEN CONDITION SET.

Action: Press **PF3** to exit the When panel.

Result: The application brings you to the previous Breakpoint Intercept panel.

Action: Continue execution by typing **GO** and pressing Enter.

Result: CAMRCOB2 resumes execution until the condition you set for the YTDCOND breakpoint is met; when the value of SUB-TOTAL is greater than or equal to 600, the application halts execution, and executes the commands you entered for the conditional breakpoint; it displays a keep window for SUB-TOTAL, sets TOTAL-GROSS to 2000, and then displays TOTAL-GROSS in the keep window. Your screen should look like the Intercept panel in the following panel.

A sample Intercept panel showing condition YTDCOND has been met follows:

```
CAMRCOB2 ---------- CA InterTest Batch WHEN YTDCOND BEFORE Intercept ---------
 COMMAND ===>                                                 SCROLL ===> CUR
 TRACE=> 001102 001101 001100 001104 001103 001102 001101 001100 001104 001103
      ------------------------------------------------------------------------
      1F08CAFC ND-OT 000679 03 SUB-TOTAL                  +0000971.68
      1F08CB08 ND-OT 000682 03 TOTAL-GROSS                +0002000.00
      ------------------------------------------------------------------------
  0003 001101           ADD BILLING-AMOUNT(SUB-7) TO SUB-TOTAL.
  0002 001102           MOVE SUB-TOTAL           TO BILL-YTD.
  0002 001103       BILL-CALC-EX.
  0002 001104           EXIT.
       001105
       001106
  ---> 001107       DEMONSTRATE-SPLIT-SCREEN.
       001108      * ---------------------------------------------*
       001109      *    This section of the Demo shows you how to   *
       001110      *    SPLIT the screen .. ..                      *
       001111      * ---------------------------------------------*
  ---> 001112           CALL 'ISPLINK' USING DSPLY,
       001113                                COBDPN73.
  ---> 001114           MOVE  4        TO    COBDPN71-LENGTHS.
  ---> 001115           CALL 'ISPLINK' USING VCOPY,
       001116                                VCOPY-COBDPN71,
```

Now remove the conditional breakpoint before continuing.

Action: Return to the When panel by entering the command: **when**

Press Enter.

Result: The When panel displays.

Action: Key in the following on the When panel:

OPTION: ===> r

WHEN NAME ===> ytdcond

Press Enter

Result: The application displays the following message in the top right corner of the When panel: WHEN DELETED.

Action: Press **PF3** to exit the When panel.

Result: The application brings you to the previous Intercept panel.

Remove the Keep Window and Continue Execution

Remove the keep window before continuing execution.

Action: Key in the following command: **remove all**

Press Enter.

Result: The keep window is removed.

Return to the Options Menu by continuing execution.

Action: Type **GO** and press Enter.

Result: CAMRCOB2 resumes execution and displays the Options Menu.

This concludes Option 2: Conditional Breakpoints. Select another option and go to the appropriate section in this guide, or exit the demo using **PF3**.

Option 3: Update Source Code

This option illustrates how you can use the ISPF or CA Roscoe split-screen capability from any screen. One way to use this feature is to update your source code as soon as you find errors during your test session.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

Action: Key in Option **3**.

Press Enter.

Result: CAMRCOB2 displays the screen that describes what occurs in this part of the demo session.

Action: Press Enter.

Result: The application halts CAMRCOB2 at an unconditional breakpoint at the paragraph SPLIT-SCREEN.

Action: ISPF users:

Tab about halfway down your screen and press the SPLIT PF key defined in your ISPF profile (normally PF2).

CA Roscoe users:

Enter the CA Roscoe SPLIT sequence.

Result: ISPF users:

The ISPF Primary Option Menu displays in the bottom half of your screen.

CA Roscoe users:

A second CA Roscoe panel displays.

A sample showing using a split-screen from an Intercept Panel (ISPF Users) follows:

```
CAMRCOB2 ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
 COMMAND ===>                                                 SCROLL ===> CUR
 TRACE=> 001123 001121 001115 001114 001112 001107 001021 001020 001018 001016
  ---> 001122              GO TO OPTIONS.
 U---> 001123        SPLIT-SCREEN.
  ---> 001124            MOVE  R1498-TIMEO TO R1498-TIME.
  ---> 001125            GO TO OPTIONS.
       001126       * ---------------------------------------------*
       001127       * From this Unconditional Breakpoint you can    *
       001128       * press the PF key assigned to the SPLIT command *
       001129       * which takes you to the ISPF Main Menu.         *
       001130       * ---------------------------------------------*
       001131
       001132
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
-----------------------  ISPF/PDF PRIMARY OPTION MENU    ISPF HELP - 461-1005
-----------------------        VERSION 3 RELEASE 1       --------------------
OPTION  ===>
                                                          USERID    - PHUMPHR
   0  ISPF PARMS  - Specify terminal and user parameters   TIME      - 14:37
   1  BROWSE      - Display source data or output listings TERMINAL - 3278
   2  EDIT        - Create or change source data           PF KEYS   - 24
   3  UTILITIES   - Perform utility functions              DATE(G)   - 14/12/18
   4  FOREGROUND  - Invoke lang. processors in foreground  DATE(J)   - 14.352
   5  BATCH       - Submit job for language processing
   6  COMMAND     - Enter TSO command or CLIST
   7  DIALOG TEST - Perform dialog testing
```

Update Your Source Code

If you had found an error in your program code, you can use the ISPF menu or CA Roscoe to access and edit your source code before continuing with your test session.

For example, in the basic demo, the application stopped execution of the program because it detected an S0C7 ABEND. You were able to continue execution by dynamically changing the value of Tasknum to zero. This was a one-time patch. To correct the problem, you need to update your source code so that Tasknum is properly initialized. You can do this from any Intercept panel by splitting your screen to edit your source code.

Continue Execution

Action: ISPF users:

Remove your ISPF split-screen using the command: **=x**

CA Roscoe users:

Use the CA Roscoe **END** key to end the second session.

Result: You return to a full-screen display of the Unconditional Breakpoint shown in the previous panel.

Action: Type **GO** and press Enter to continue execution.

Result: You return to the Demo Options Menu.

This concludes Option 3: Update Source Code. Select another option and go to the appropriate section in this guide, or exit the demo using **PF3**.

Option 4: Interrupt a Looping Program

This option illustrates how to interrupt and verify that you have a looping program. It also illustrates the use of the skip command.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

This option requires you to interrupt the program by pressing the **ATTENTION** key or the **PA1** key. Before proceeding, locate each of these keys on your keyboard. If you need help, ask a technical support person **before** you begin.

Action: Key in Option **4**. Press Enter.

Result: CAMRCOB2 displays the screen that describes what occurs in this part of the demo session.

Action: Press Enter.

Result: The application halts CAMRCOB2 at an unconditional breakpoint set at ORDER-CALC.

A sample screen showing the CAMRCOB2 stopped at ORDER-CALC follows:

```
CAMRCOB2 ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
 COMMAND ===>                                                 SCROLL ===> CUR
 TRACE=> 001232 001225 001219 001218 001216 001210 001023 001022 001020 001018
       001231       * ---------------------------------------------*
 U---> 001232        ORDER-CALC.
  ---> 001233            MOVE CUSTOMER-VALUES TO CUSTOMER-ORDER.
  ---> 001234            MOVE ZEROES TO ORDER-SUB-TOTAL, SUB-6, LOOP-OUT.
  ---> 001235            PERFORM TOTAL-ORDER THRU TOTAL-ORDER-EX
       001236                UNTIL SUB-6 EQUAL 5.
 U---> 001237        ORDER-CALC-EX.
  ---> 001238            GO TO OPTIONS.
  ---> 001239        TOTAL-ORDER.
  ---> 001240            ADD 1 TO SUB-6.
  ---> 001241            ADD 1 TO LOOP-OUT.
  ---> 001242            MULTIPLY UNIT-PRICE(SUB-6) BY NO-OF-WIDGETS(SUB-6)
       001243                GIVING ITEM-TOTAL.
  ---> 001244            ADD ITEM-TOTAL TO ORDER-SUB-TOTAL.
  ---> 001245            MOVE 1 TO SUB-6.
  ---> 001246        TOTAL-ORDER-EX.
       001247       *---------------------------------------------*
       001248       * The code above will loop. The loop can be      *
```

Note: If the Frequency Counter arrows and numbers are not displayed along the left side of your screen, turn the frequency display on by entering the command: **freq**.

Action: Continue execution by typing **GO** and pressing Enter.

Result: CAMRCOB2 resumes execution.

Notice the system light remains on. There are additional breakpoints and screen panels in the demo program code, yet all you see is the system light. This is symptomatic of a looping program. Halt the program to see what is going on.

Action: Press the **ATTN** or **PA1** key. Try the **ATTN** key first. If that does not work, your testing environment requires you to use the **PA1** key. Press **RESET** before pressing the **PA1** key to unlock the terminal keyboard.

Result: The Attention Intercept panel displays.

Continue with the Continuing from the Attention Routine section.

Note: If you do not successfully stop execution with the ATTN or PA1 key before 5000 executions of the demo program loop, you reach a safety net conditional breakpoint. If you reach this breakpoint, instructions for continuing are given in the Continuing from the "Safety Net" Conditional Breakpoint section.

Continue from the "Safety Net" Conditional Breakpoint

Follow this section only if you see the breakpoint in the following panel. If you do not, continue with the Continuing from the Attention Routine section.

A sample screen showing a Safety Net Conditional Breakpoint of a Demo Loop follows:

```
CAMRCOB2 ---------- CA InterTest Batch WHEN LOOPCOND BEFORE Inter- SET COMPLETE
 COMMAND ===>                                                  SCROLL ===> CUR
 TRACE=> 001242 001241 001240 001239 001253 001246 001245 001244 001242 001241
      001231       * ---------------------------------------------*
 U0001 001232       ORDER-CALC.
  0001 001233           MOVE CUSTOMER-VALUES TO CUSTOMER-ORDER.
  0001 001234           MOVE ZEROES TO ORDER-SUB-TOTAL, SUB-6, LOOP-OUT.
  0001 001235           PERFORM TOTAL-ORDER THRU TOTAL-ORDER-EX
      001236                UNTIL SUB-6 EQUAL 5.
 U---> 001237       ORDER-CALC-EX.
  ---> 001238           GO TO OPTIONS.
  5001 001239       TOTAL-ORDER.
  5001 001240           ADD 1 TO SUB-6.
  5001 001241           ADD 1 TO LOOP-OUT.
  5000 001242           MULTIPLY UNIT-PRICE(SUB-6) BY NO-OF-WIDGETS(SUB-6)
      001243                GIVING ITEM-TOTAL.
  5000 001244           ADD ITEM-TOTAL TO ORDER-SUB-TOTAL.
  5000 001245           MOVE 1 TO SUB-6.
  5000 001246       TOTAL-ORDER-EX.
      001247       *---------------------------------------------*
```

To continue with the looping program demo, you must press the **ATTN** or **PA1 key** before 5000 executions of the loop this time. To continue the program loop, use the following steps:

1. Type a **g** to the left of the line:

   MOVE ZEROES TO ORDER-SUB-TOTAL, SUB-6, LOOP-0

2. Press Enter to continue the loop execution.

3. Press the **ATTN** key or the **RESET**, then **PA1** keys. Try the **ATTN** key first. If that does not work, your testing environment requires you to use the **PA1** key.

Note: Even if **PA1** stopped the slow execution in option 1, you may need to press **ATTN** to stop a loop.

You should see the Attention Intercept panel.

To return to the Options Menu:

1. Enter the command: **go options**

2. Press Enter.

   You return to the Options Menu. You can try the same option again, even if you did not complete it the first time, or select another option from the menu.

Continue from the Attention Routine

The Attention Intercept panel displays. A sample showing an Attention Intercept in TOTAL-ORDER Routine follows:

```
CAMRCOB2 ----------- CA InterTest Batch ATTENTION INTERCEPT -------------------
COMMAND ===>                                                 SCROLL ===> PAGE
TRACE=> 001244 001243 001242 001241 001255 001248 001247 001246 001244 001243 0
       001231       * ---------------------------------------------*
 U0001 001232        ORDER-CALC.
  0001 001233            MOVE CUSTOMER-VALUES TO CUSTOMER-ORDER.
  0001 001234            MOVE ZEROES TO ORDER-SUB-TOTAL, SUB-6, LOOP-OUT.
  0001 001235            PERFORM TOTAL-ORDER THRU TOTAL-ORDER-EX
       001236                UNTIL SUB-6 EQUAL 5.
 U---> 001237        ORDER-CALC-EX.
  ---> 001238            GO TO OPTIONS.
  1926 001239        TOTAL-ORDER.
  1926 001240            ADD 1 TO SUB-6.
  1926 001241            ADD 1 TO LOOP-OUT.
  1925 001242            MULTIPLY UNIT-PRICE(SUB-6) BY NO-OF-WIDGETS(SUB-6)
       001243                GIVING ITEM-TOTAL.
  1925 001244            ADD ITEM-TOTAL TO ORDER-SUB-TOTAL.
  1925 001245            MOVE 1 TO SUB-6.
  1925 001246        TOTAL-ORDER-EX.
       001247       *---------------------------------------------*
```

CAMRCOB2 never left the TOTAL-ORDER routine, as you can see from your Attention Intercept screen display. The current statement is highlighted.

Look at the frequency counters to the left of the program statements. They show repeated execution of the TOTAL-ORDER statements, indicating a loop. To investigate the logic in TOTAL-ORDER, you can set a keep window for SUB-6 and then execute slowly using the SLOW command. This time you use the abbreviated form of the keep command: **k**.

Action: Key in the following command to set the keep window: **k sub-6**

Press Enter.

Result: The application displays a keep window for the variable SUB-6.

A sample screen showing the keep window to view SUB-6 follows:

```
CAMRCOB2 ----------- CA InterTest Batch ATTENTION INTERCEPT -----------------
COMMAND ===>                                                 SCROLL ===> PAGE
TRACE=> 001244 001243 001242 001241 001255 001248 001247 001246 001244 001243 0
     ------------------------------------------------------------------------
     001424D8 NB-S  000046 77 SUB-6                    +0002.
     ------------------------------------------------------------------------
      001231       * ---------------------------------------------*
 U0001 001232        ORDER-CALC.
  0001 001233            MOVE CUSTOMER-VALUES TO CUSTOMER-ORDER.
  0001 001234            MOVE ZEROES TO ORDER-SUB-TOTAL, SUB-6, LOOP-OUT.
  0001 001235            PERFORM TOTAL-ORDER THRU TOTAL-ORDER-EX
      001236                UNTIL SUB-6 EQUAL 5.
 U---> 001237        ORDER-CALC-EX.
  ---> 001238            GO TO OPTIONS.
  1926 001239        TOTAL-ORDER.
  1926 001240            ADD 1 TO SUB-6.
  1926 001241            ADD 1 TO LOOP-OUT.
  1925 001242            MULTIPLY UNIT-PRICE(SUB-6) BY NO-OF-WIDGETS(SUB-6)
      001243                GIVING ITEM-TOTAL.
  1925 001244            ADD ITEM-TOTAL TO ORDER-SUB-TOTAL.
  1925 001245            MOVE 1 TO SUB-6.
  1925 001246        TOTAL-ORDER-EX.
```

Action: Key in the following command to execute CAMRCOB2 one verb every two seconds: **slow 2**

Press Enter.

Result: CAMRCOB2 continues a slow execution. As the code executes, you can see the code looping, and you can see the corresponding value of SUB-6 in the keep window.

The following sample screen of a SLOW 2 Execution shows a Loop and SUB-6 problem:

```
CAMRCOB2 ---------- CA InterTest Batch STEP BEFORE Intercept -----------------
 COMMAND ===>                                                 SCROLL ===> CUR
 TRACE=> 001244 001242 001241 001240 001239 001253 001246 001245 001244 001242
      ------------------------------------------------------------------------
      1F08D1C8 NB-S  000044 77 SUB-6                      +0002.
      ------------------------------------------------------------------------
      001231       * ---------------------------------------------*
 U0001 001232       ORDER-CALC.
  0001 001233           MOVE CUSTOMER-VALUES TO CUSTOMER-ORDER.
  0001 001234           MOVE ZEROES TO ORDER-SUB-TOTAL, SUB-6, LOOP-OUT.
  0001 001235           PERFORM TOTAL-ORDER THRU TOTAL-ORDER-EX
      001236               UNTIL SUB-6 EQUAL 5.
 U---> 001237       ORDER-CALC-EX.
  ---> 001238           GO TO OPTIONS.
  5002 001239       TOTAL-ORDER.
  5002 001240           ADD 1 TO SUB-6.
  5002 001241           ADD 1 TO LOOP-OUT.
  5002 001242           MULTIPLY UNIT-PRICE(SUB-6) BY NO-OF-WIDGETS(SUB-6)
      001243               GIVING ITEM-TOTAL.
  5001 001244           ADD ITEM-TOTAL TO ORDER-SUB-TOTAL.
  5001 001245           MOVE 1 TO SUB-6.
  5001 001246       TOTAL-ORDER-EX.
      001247       *---------------------------------------------*
      001248       * The code above will loop. The loop can be      *
```

Notice that SUB-6 changes from 1 to 2, but back to 1 again. SUB-6 is being reset to 1, and causing the program to loop. The value of SUB-6 is preventing the exit of the TOTAL-ORDER routine.

Action: Halt execution again by pressing the ATTENTION or PA1 key.

After the Attention Routine message displays, press Enter.

Result: The application brings you to another Attention Intercept panel. The current statement is highlighted.

To dynamically correct the loop-condition, you could use the SKIP command to bypass the offending statement. The **s** line command is another form of the SKIP command.

Action: Tab down to the statement: MOVE 1 TO SUB-6, and key an **s** to the left of the line.

The following sample screen shows using Skip to bypass the cause of the loop:

```
CAMRCOB2 ---------- CA InterTest Batch ATTENTION Intercept ------------------
COMMAND ===>                                                 SCROLL ===> CUR
 TRACE=> 001244 001242 001241 001240 001239 001253 001246 001245 001244 001242
      ------------------------------------------------------------------------
      1F08D1C8 NB-S  000044 77 SUB-6                      +0002.
      ------------------------------------------------------------------------
      001231       * ---------------------------------------------*
 U0001 001232        ORDER-CALC.
  0001 001233            MOVE CUSTOMER-VALUES TO CUSTOMER-ORDER.
  0001 001234            MOVE ZEROES TO ORDER-SUB-TOTAL, SUB-6, LOOP-OUT.
  0001 001235            PERFORM TOTAL-ORDER THRU TOTAL-ORDER-EX
      001236                UNTIL SUB-6 EQUAL 5.
 U---> 001237        ORDER-CALC-EX.
  ---> 001238            GO TO OPTIONS.
  5002 001239        TOTAL-ORDER.
  5002 001240            ADD 1 TO SUB-6.
  5002 001241            ADD 1 TO LOOP-OUT.
  5002 001242            MULTIPLY UNIT-PRICE(SUB-6) BY NO-OF-WIDGETS(SUB-6)
      001243                GIVING ITEM-TOTAL.
  5001 001244            ADD ITEM-TOTAL TO ORDER-SUB-TOTAL.
 s5001 001245            MOVE 1 TO SUB-6.
  5001 001246        TOTAL-ORDER-EX.
      001247       *---------------------------------------------*
      001248       * The code above will loop. The loop can be      *
```

Result: The application skips the execution of the MOVE statement, and executes the code as if the MOVE statement was removed.

Action: Tab up to the first statement in the TOTAL-ORDER routine, ADD 1 to SUB-6, and type a **g** to the left of the line, to continue execution from that point.

Your entry should match the following panel:

The following sample screen shows a continuing execution from the ADD Statement:

```
CAMRCOB2 ---------- CA InterTest Batch ATTENTION Intercept ------------------
COMMAND ===>                                                 SCROLL ===> CUR
 TRACE=> 001244 001242 001241 001240 001239 001253 001246 001245 001244 001242
       ------------------------------------------------------------------------
       1F08D1C8 NB-S  000044 77 SUB-6                      +0002.
       ------------------------------------------------------------------------
       001231       * ---------------------------------------------*
 U0001 001232        ORDER-CALC.
  0001 001233            MOVE CUSTOMER-VALUES TO CUSTOMER-ORDER.
  0001 001234            MOVE ZEROES TO ORDER-SUB-TOTAL, SUB-6, LOOP-OUT.
  0001 001235            PERFORM TOTAL-ORDER THRU TOTAL-ORDER-EX
       001236                UNTIL SUB-6 EQUAL 5.
 U---> 001237        ORDER-CALC-EX.
  ---> 001238            GO TO OPTIONS.
  5002 001239        TOTAL-ORDER.
 g5002 001240            ADD 1 TO SUB-6.
  5002 001241            ADD 1 TO LOOP-OUT.
  5002 001242            MULTIPLY UNIT-PRICE(SUB-6) BY NO-OF-WIDGETS(SUB-6)
       001243                GIVING ITEM-TOTAL.
  5001 001244            ADD ITEM-TOTAL TO ORDER-SUB-TOTAL.
 S5001 001245            MOVE 1 TO SUB-6.
  5001 001246        TOTAL-ORDER-EX.
       001247       *---------------------------------------------*
       001248       * The code above will loop. The loop can be      *
```

Press Enter.

Result: CAMRCOB2 resumes execution from the line where you entered the g, and continues execution until it reaches the next unconditional breakpoint, as shown in the following panel.

The following screen shows an unconditional breakpoint at ORDER-CALC-EX:

```
CAMRCOB2 ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
 COMMAND ===>                                                 SCROLL ===> CUR
 TRACE=> 001237 001253 001246 001244 001242 001241 001240 001239 001253 001246
       ------------------------------------------------------------------------
       1F08D1C8 NB-S  000044 77 SUB-6                      +0005.
       ------------------------------------------------------------------------
       001231       * ---------------------------------------------*
 U0001 001232        ORDER-CALC.
  0001 001233            MOVE CUSTOMER-VALUES TO CUSTOMER-ORDER.
  0001 001234            MOVE ZEROES TO ORDER-SUB-TOTAL, SUB-6, LOOP-OUT.
  0001 001235            PERFORM TOTAL-ORDER THRU TOTAL-ORDER-EX
       001236                UNTIL SUB-6 EQUAL 5.
 U---> 001237        ORDER-CALC-EX.
  ---> 001238            GO TO OPTIONS.
  5004 001239        TOTAL-ORDER.
  5005 001240            ADD 1 TO SUB-6.
  5005 001241            ADD 1 TO LOOP-OUT.
  5005 001242            MULTIPLY UNIT-PRICE(SUB-6) BY NO-OF-WIDGETS(SUB-6)
       001243                GIVING ITEM-TOTAL.
  5005 001244            ADD ITEM-TOTAL TO ORDER-SUB-TOTAL.
 S5001 001245            MOVE 1 TO SUB-6.
  5004 001246        TOTAL-ORDER-EX.
       001247       *---------------------------------------------*
       001248       * The code above will loop. The loop can be      *
```

CAMRCOB2 finally executed out of the TOTAL-ORDER routine, and continued until it reached unconditional breakpoint set at ORDER-CALC-EX (one of the unconditional breakpoints included in the data set member demoincl).

You can see from the keep window that the value of SUB-6 is 5. This is the proper value to permit the exit of the ORDER-CALC routine. Thus, the use of the skip command allowed us to dynamically fix the program code that caused the loop, and continue execution.

Remove the Keep Window and Skip Command

Before going back to the options menu, remove the keep window and skip command as follows.

Action: Key in the command: **remove all**

Tab down and type over the S to the left of the MOVE statement with an **x**.

Press Enter.

Result: The keep window is removed, and the skip line command is removed.

Action: Continue execution by typing **GO** and pressing Enter.

Result: CAMRCOB2 resumes execution and displays the Demo Session Options Menu.

This concludes Option 4: Interrupt a Program Loop. Select another option and go to the appropriate section in this guide, or exit the demo using **PF3**.

Option 5: Trace Program Execution

This option illustrates the use of the PREVIOUS and ADVANCE commands to help you trace program execution.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

Action: Key in Option **5**.

Press Enter.

Result: CAMRCOB2 displays the screen that describes what occurs in this part of the demo session.

Action: Press Enter.

Result: The application halts CAMRCOB2 at an unconditional breakpoint that you included at the beginning of the demo.

The following sample screen shows CAMRCOB2 at an unconditional breakpoint:

```
CAMRCOB2 ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
 COMMAND ===>                                                 SCROLL ===> CUR
 TRACE=> 001055 001158 001157 001156 001155 001153 001147 001146 001139 001137
       001054       *
 U---> 001055        PAY-CALC.
  ---> 001056            PERFORM POPULATE-GRAPH THRU POPULATE-GRAPH-EX
       001057                VARYING SUB-4 FROM 1 BY 1
       001058                    UNTIL SUB-4 GREATER THAN NO-OF-HOURS.
  ---> 001059        PAY-CALC-EX.
  ---> 001060            MOVE SPACES      TO CHEQUE-LINE.
  ---> 001061            MOVE TOTAL-GROSS TO CHEQUE-AMOUNT.
  ---> 001062        PAY-RETURN.
  ---> 001063            GO TO OPTIONS.
       001064       *
  ---> 001065        POPULATE-GRAPH.
  ---> 001066            MOVE MONTH-ITEM(SUB-4)     TO X-AXIS-YTD.
  ---> 001067            MOVE TOKEN-ITEM            TO X-AXIS-R(SUB-4).
  ---> 001068            ADD  MONTHLY-AMOUNT(SUB-4) TO YTD.
  ---> 001069            MOVE YTD                   TO TOTAL-GROSS.
  ---> 001070        POPULATE-GRAPH-EX.
  ---> 001071            EXIT.
  ---> 001072        DEMONSTRATE-CONDITIONAL-BKPT.
  ---> 001073            CALL 'ISPLINK' USING DSPLY,
```

Suppose you want to trace the execution path of the program prior to this point. You can have the application scroll back and display a previously executed statement using the PREV command. The **PREV** command lets you view previously executed statements, but does not cause any statements to execute. After you scroll back using the PREV command, you can scroll forward and retrace execution using the **ADVANCE** command. You may specify a number from 1 to 999 with the PREV and ADVANCE commands to scroll more than one statement at a time.

The PREV and ADVANCE commands do not change the statement where CAMRCOB2 resumes execution; the PREV and ADVANCE only scroll the statements in the trace.

Action: Key in the following in the Command field: **prev 9**

Press Enter.

Result: The application displays the 9th previous statement executed.

The following screen shows the ninth previous statement:

```
CAMRCOB2 ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
 COMMAND ===>                                                 SCROLL ===> CUR
 TRACE=> 001055 001158 001157 001156 001155 001153 001147 001146 001139 001137
      001136
  0001 001137        DEMONSTRATE-PREV-COMMAND.
      001138       *
  0001 001139            CALL 'ISPLINK' USING DSPLY,
      001140                                 COBDPN75.
      001141       * ---------------------------------------------*
      001142       * This section of the Demo shows you           *
      001143       * the benefits of using the Prev               *
      001144       * and Advance commands.. ..                    *
      001145       * ---------------------------------------------*
  0001 001146            MOVE  4        TO    COBDPN71-LENGTHS.
  0001 001147            CALL 'ISPLINK' USING VCOPY,
      001148                                 VCOPY-COBDPN71,
      001149                                 VCOPY-COBDPN71-LENGTHS,
      001150                                 VCOPY-COBDPN71-VALUES,
      001151                                 VCOPY-MOVE.
      001152       *
  0001 001153            IF VCOPY-COBDPN71-EIBAID = 'PF03' OR 'PF15'
  ---> 001154               GO TO OPTIONS.
  0001 001155            MOVE SPACES  TO X-AXIS.
```

Now you can retrace the execution of the nine statements using the ADVANCE command. The short form of the ADVANCE command is A or AD. (You must enter the A on the command line.)

Action: Key in the command: **a**

Press Enter.

Result: The application highlights the next statement executed.

Action: Repeat the Advance step a few more times.

The following screen shows advancing through the program trace:

```
CAMRCOB2 ---------- CA InterTest Batch UNCOND BEFORE Intercept --------------
 COMMAND ===>                                                  SCROLL ===> CUR
 TRACE=> 001055 001158 001157 001156 001155 001153 001147 001146 001139 001137
      001136
  0001 001137        DEMONSTRATE-PREV-COMMAND.
      001138       *
  0001 001139            CALL 'ISPLINK' USING DSPLY,
      001140                                 COBDPN75.
      001141       * ---------------------------------------------*
      001142       * This section of the Demo shows you           *
      001143       * the benefits of using the Prev               *
      001144       * and Advance commands.. ..                    *
      001145       * ---------------------------------------------*
  0001 001146            MOVE  4        TO    COBDPN71-LENGTHS.
  0001 001147            CALL 'ISPLINK' USING VCOPY,
      001148                                 VCOPY-COBDPN71,
      001149                                 VCOPY-COBDPN71-LENGTHS,
      001150                                 VCOPY-COBDPN71-VALUES,
      001151                                 VCOPY-MOVE.
      001152       *
  0001 001153            IF VCOPY-COBDPN71-EIBAID = 'PF03' OR 'PF15'
  ---> 001154               GO TO OPTIONS.
  0001 001155            MOVE SPACES  TO X-AXIS.
```

Result: You can follow the execution order of the program by following the highlighted lines as you advance through the program trace. In effect, using PREV and ADVANCE gives you a simulated instant replay of program execution.

Return to the Current Statement

Now you can return to the current statement, that is, the statement where execution was halted prior to the unconditional breakpoint.

Action: Enter the command: **cs**

Press Enter.

Result: The application displays the original Breakpoint Intercept panel shown in the beginning of this option.

Action: Key in the command: **go**

Press Enter

Result: CAMRCOB2 resumes execution and displays the Advanced Options Demo Screen. You can continue with any option or exit.

This concludes Option 5: Trace Program Execution. Select another option and continue with the appropriate section in this guide, or press **PF3** to end the demo.

Option 6: Work with Indexed Table Items

This section of the demo session shows you a fast and easy way to inspect and modify the contents of indexed data items. It also demonstrates how to use the STEP command to single-step execution.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

Action: Select Option **6**.

Press Enter.

Result: CAMRCOB2 displays a screen that describes what occurs in this part of the demo session.

Action: Press Enter.

Result: CAMRCOB2 resumes execution.

The next screen you see is an Abend Intercept panel. The application has automatically halted CAMRCOB2 because it detected a data exception in the highlighted ADD instruction.

The following screen shows the CAMRCOB2 at an S0C7 Abend Intercept:

```
CAMRCOB2 ---------- CA InterTest Batch ABEND S0C7 Intercept ------------------
 COMMAND ===>                                                 SCROLL ===> CUR
 TRACE=> 001184 001183 001182 001181 001196 001195 001193 001192 001196 001195
  0001 001183           SET C1 TO 5.
  0001 001184           ADD +1 TO DISTRICT ( A1, B1, C1 ).
  ---> 001185           MOVE SPACES TO TASK-TEXT OF TASK-STRUCTURE.
       001186       * ---------------------------------------------*
       001187       * This section of the Demo shows you how to     *
       001188       * inspect and change the value of subscripted   *
       001189       * table items.. ..                              *
       001190       * ---------------------------------------------*
  ---> 001191           GO TO OPTIONS.
       PP 5655-G53 IBM Enterprise COBOL for z/OS and OS/390 3.1.0     CAMRCOB2
       LineID ----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+
  0018 001192        INITIALIZE-TABLE.
  0018 001193           MOVE SUB-1 TO STATE-NUMBER (SUB-1),
       001194                         COUNTY-NUM1 (SUB-1, SUB-2).
  0018 001195           MOVE ','   TO COUNTY-COMM (SUB-1, SUB-2).
  0018 001196           MOVE SUB-2 TO COUNTY-NUM2 (SUB-1, SUB-2).
  ---> 001197        SET-VAR-REC.
  ---> 001198           MOVE +100 TO VAR-REC-LEN.
  ---> 001199           MOVE ZERO TO VAR-SS.
  ---> 001200        PROCESS-VAR.
```

If you recall, this is the same type of error detected and corrected in the basic demo session. To continue execution, you need to find the data item causing the S0C7 ABEND and dynamically modify it.

Looking at the highlighted breakpoint statement, you might suspect the variable is at fault; it probably was not initialized correctly. The variable in this case is an indexed table entry.

Display an Indexed Table Entry

You can quickly inspect the value of the **DISTRICT (A1, B1, C1)** field directly from this screen by entering a **k** to the left of the highlighted statement.

Action: Tab to the highlighted ADD instruction and type **k** (for keep).

Press Enter.

Result: The application displays the current value of DISTRICT. It also displays the value of each subscripted index item, because the k line command requests a display of each item on that line.

The following panel shows displaying a keep window for the Table Entry:

```
CAMRCOB2 ---------- CA InterTest Batch ABEND S0C7 Intercept -----------------
 COMMAND ===>                                                SCROLL ===> CUR
 TRACE=> 001184 001183 001182 001181 001196 001195 001193 001192 001196 001195
      ------------------------------------------------------------------------
      1F08E5BE NP-S  000856 07 DISTRICT                   ?000.
                                A1  (+1)
                                B1  (+3)
                                C1  (+5)
      ------------------------------------------------------------------------
  0001 001183           SET C1 TO 5.
  0001 001184           ADD +1 TO DISTRICT ( A1, B1, C1 ).
  ---> 001185           MOVE SPACES TO TASK-TEXT OF TASK-STRUCTURE.
       001186       * ---------------------------------------------*
       001187       * This section of the Demo shows you how to     *
       001188       * inspect and change the value of subscripted   *
       001189       * table items.. ..                              *
       001190       * ---------------------------------------------*
  ---> 001191           GO TO OPTIONS.
       PP 5655-G53 IBM Enterprise COBOL for z/OS and OS/390 3.1.0     CAMRCOB2
       LineID ----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+
  0018 001192        INITIALIZE-TABLE.
  0018 001193           MOVE SUB-1 TO STATE-NUMBER (SUB-1),
```

The question mark that precedes the value of DISTRICT in the keep window tells us this field does not contain a valid value. This is the error that triggered the automatic breakpoint.

The keep window also tells us which table entry is at fault. The values of each index item in this window indicate the invalid data occurs at element (1, 3, 5).

Dynamically Correct an Uninitialized Table Item

To dynamically correct this error you need to modify main storage so the field contains valid data. You can use the SET command to set the value to zero. The application sets the variable using the proper format.

Action: Type the command:

set district (a1,b1,c1) = zero

Press Enter.

The following panel shows correcting the uninitialized Table Item:

```
CAMRCOB2 ---------- CA InterTest Batch ABEND S0C7 Intercept -----------------
 COMMAND ===> set district (a1,b1,c1) = zero                  SCROLL ===> CUR
 TRACE=> 001184 001183 001182 001181 001196 001195 001193 001192 001196 001195
      ------------------------------------------------------------------------
      1F08E5BE NP-S  000856 07 DISTRICT                    ?000.
                                A1  (+1)
                                B1  (+3)
                                C1  (+5)
      ------------------------------------------------------------------------
  0001 001183           SET C1 TO 5.
  0001 001184           ADD +1 TO DISTRICT ( A1, B1, C1 ).
  ---> 001185           MOVE SPACES TO TASK-TEXT OF TASK-STRUCTURE.
       001186       * -------------------------------------------*
       001187       * This section of the Demo shows you how to  *
       001188       * inspect and change the value of subscripted *
       001189       * table items.. ..                           *
       001190       * -------------------------------------------*
  ---> 001191           GO TO OPTIONS.
       PP 5655-G53 IBM Enterprise COBOL for z/OS and OS/390 3.1.0     CAMRCOB2
       LineID ----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+
  0018 001192        INITIALIZE-TABLE.
  0018 001193           MOVE SUB-1 TO STATE-NUMBER (SUB-1),
```

Result: The application dynamically modifies the value in main storage for DISTRICT (A1, B1, C1), and displays the new value in the keep window. The message on the top right of the panel reads: **SET COMPLETE**.

Single-Step Execution

Use the step command to execute just one line of code. The step command lets you determine how many lines are executed when you use the go command. For example, to execute one line at a time, or single-step, use the command: **step 1**.

Action: Type the command: **step 1**

Press Enter.

Result: The application sets the step count to one and displays the following message: **STEP COUNT SET**.

The following panel shows setting the step count:

```
CAMRCOB2 ---------- CA InterTest Batch ABEND S0C7 Intercept ---- STEP COUNT SET
 COMMAND ===>                                                   SCROLL ===> CUR
 TRACE=> 001184 001183 001182 001181 001196 001195 001193 001192 001196 001195
      ------------------------------------------------------------------------
      1F08E5BE NP-S  000856 07 DISTRICT                    +000.
                                A1  (+1)
                                B1  (+3)
                                C1  (+5)
      ------------------------------------------------------------------------
  0001 001183           SET C1 TO 5.
  0001 001184           ADD +1 TO DISTRICT ( A1, B1, C1 ).
  ---> 001185           MOVE SPACES TO TASK-TEXT OF TASK-STRUCTURE.
       001186       * ---------------------------------------------*
       001187       * This section of the Demo shows you how to     *
       001188       * inspect and change the value of subscripted   *
       001189       * table items.. ..                              *
       001190       * ---------------------------------------------*
  ---> 001191           GO TO OPTIONS.
       PP 5655-G53 IBM Enterprise COBOL for z/OS and OS/390 3.1.0     CAMRCOB2
       LineID ----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+
  0018 001192        INITIALIZE-TABLE.
  0018 001193           MOVE SUB-1 TO STATE-NUMBER (SUB-1),
```

Action: Execute CAMRCOB2 by entering the command: **GO** and pressing Enter.

Result: The ADD statement is executed, and then the application halts execution and displays a Step Count Intercept panel. The line below the ADD statement is now the current line, and is highlighted.

Remove the Step Count and Keep Window

When you want to end single-stepping, reset the step count to zero. This turns single stepping off. Remove the step count and keep window before continuing execution.

Action: Type in the command:

step 0; remove all

Press Enter.

Result: The application resets the step count and removes the keep window.

To display the current status of the step count, enter the step command without any count number as follows:

Action: Type in the command: **step**

Press Enter.

Result: The application displays the value of the step count in the top-right corner; in this case, the message is: NO STEP COUNT SET.

Action: Resume execution of CAMRCOB2 by entering **go**.

Result: CAMRCOB2 resumes execution, and you return to the Options Menu.

This concludes Option 6: Work with Indexed Table Items. Select another option and continue with the appropriate section in this guide, or press **PF3** to end the demo.

Option 7: Histogram Report

This option illustrates how to create and view the Histogram Report of Statement Execution, which is useful in identifying dead code.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

Action: Key in Option **7**.

Press Enter.

Result: CAMRCOB2 displays the screen that describes what occurs in this part of the demo session.

Action: Press Enter.

Result: The application halts CAMRCOB2 at an unconditional breakpoint set at the program exit. The PGM Exit Intercept panel displays in the following panel.

The following screen shows the Program Exit Breakthrough Intercept panel:

```
CAMRCOB2 ---------- CA InterTest Batch PGM EXIT Intercept --------------------
 COMMAND ===>                                                 SCROLL ===> CSR
 TRACE=> 000975 000974 000973 001277 001310 001309 001308 001318 001317 001316
  0001 000974           CLOSE REPORT-OUT.
  ---> 000975           GOBACK.
       000976
       000977       * ---------------------------------------------*
       000978       * The " UNCOND ALL EXIT " ( or " AT ALL EXIT " ) *
       000979       * command will cause an Unconditional Breakpoint *
       000980       * at a program's exit point.. ..                 *
       000981       * ---------------------------------------------*
       000982
  0001 000983        OPTIONS-MENU.
       000984       * ---------------------------------------------*
       000985       * Initialise the lengths in case of recursion .. *
       000986       * ---------------------------------------------*
  0001 000987           CALL 'ISPLINK' USING DSPLY,
       000988                                COBDPN70.
       000989
  0001 000990           MOVE 4         TO OPTION-OPTAID.
  0001 000991           CALL 'ISPLINK' USING VCOPY,
       000992                                VCOPY-OPTION-LIST,
       000993                                VCOPY-OPTION-LENGTHS,
```

This type of unconditional breakpoint is set with the command: UNCOND ALL EXIT or AT ALL EXIT. Prior to exiting the program, you can request a Histogram Report using the HIST command.

Action: Key in the command: **hist**

Press Enter.

Result: The application returns the following message:

'HISTOGRAM SUCCESSFULLY WRITTEN'

View the Histogram Report

The report is written to the data set associated with the ddname INT1REPT that you can view by splitting the screen.

Action: ISPF users:

Tab a few lines down your screen and press the SPLIT PF key defined in your ISPF profile (normally PF2).

CA Roscoe users:

Enter the CA Roscoe SPLIT sequence.

Result: ISPF users:
The ISPF Main Menu displays in the split-screen area.

CA Roscoe users:
A second CA Roscoe panel displays.

Action: ISPF users:
Select the ISPF Browse Facility and request a display of the data set associated with the ddname INT1REPT. You allocated this data set in the Advanced Demo Preliminaries section The suggested data set name was: userid.INT1REPT

CA Roscoe users:
Use CA Roscoe Attach DSN command to display the data set member associated with the ddname INT1REPT. The suggested data set name was: userid.INT1REPT .

Note: In the following example, the data set name is: intbatch.histrept.

Result: The first page of the histogram report displays. A sample report is shown in the following panel.

The following screen shows Browsing the Histogram Dataset Member:

```
CAMRCOB2 ---------- CA InterTest Batch PGM EXIT- HISTOGRAM SUCCESSFULLY WRITTEN
 COMMAND ===>                                                 SCROLL ===> CUR
 TRACE=> 000975 000974 000973 001277 001310 001309 001308 001318 001317 001316
  0001 000974            CLOSE REPORT-OUT.
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 BROWSE -- USER99 _TSU04928 ------------------------ Line 00000000 Col 001 080
 Command ===>                                                 Scroll ===> PAGE
******************************** Top of Data ********************************
1CA InterTest/Batch       Execution Histogram For Program-ID: CAMRCOB2     Date
0
 000876 0001 |*
 000877 0001 |*
 000887 0001 |*
 000888 0001 |*
 000890 0001 |*
 000891 0001 |*
 000892 0001 |*
 000896 0002 |**
 000897 0002 |**
 000898 ---- |
 000900 0002 |**
 000901 0002 |**
 000902 0001 |*
```

Page down to the end of the report using the maximum command with PF8.

Action: Type in the command: **m**

Press PF8.

Result: The last page of the Histogram Report displays, as shown in the following panel.

The following screen shows the last page of Histogram Report:

```
CAMRCOB2 ---------- CA InterTest Batch PGM EXIT- HISTOGRAM SUCCESSFULLY WRITTEN
 COMMAND ===>                                                 SCROLL ===> CUR
 TRACE=> 000975 000974 000973 001277 001310 001309 001308 001318 001317 001316
  0001 000974           CLOSE REPORT-OUT.
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 BROWSE -- USER99 _TSU04928 ------------------------ Line 00000246 Col 001 080
 Command ===>                                                 Scroll ===> PAGE
 001310 0017 |*****************
 001311 ---- |
 001312 ---- |
 001313 ---- |
 001314 0034 |**********************************
 001315 0034 |**********************************
 001316 0034 |**********************************
 001317 0034 |**********************************
 001318 0034 |**********************************
              ----+----|----+----|----+----|----+----|----+----|----+----|----+
0            *NOTE* ASTERISKS REPRESENT A SCALE OF 00000001 EXECUTION(S) PER AS
1CA InterTest/Batch       Execution Histogram For Program-ID: CAMRCOB2     Date
-Session Number: 1,175For Userid USER99
-Number of executable statements: 236
0Number of statements executed during this test: 116
0Percentage of statements executed during this test: 49
```

In the previous panel, you can see that a number of lines were never executed. Check to see if these lines in the program are dead code and should be removed.

Note: In our example, the unexecuted code starts at line 1311. Note the corresponding line number in your program that is unexecuted.

Action: ISPF users:

Remove your split-screen using the command: **x**

CA Roscoe users:

Use the CA Roscoe **END** key to end the second session.

Result: You return to a full-screen display of the Program Exit Intercept shown in the beginning of this option.

Use the Find Statement (FS) command to display the code that was never executed.

Action: Type in the command:

fs 1311

Press Enter.

Result: The application displays the requested line of code.

The following screen displays the unexecuted code:

```
CAMRCOB2 ---------- CA InterTest Batch PGM EXIT Intercept --------------------
 COMMAND ===>                                                 SCROLL ===> CUR
 TRACE=> 000975 000974 000973 001277 001310 001309 001308 001318 001317 001316
  ---> 001311        990-INITIALISE.
  ---> 001312            MOVE SPACES TO DM-TEXT.
  ---> 001313            MOVE ZEROES TO DM-RC.
  0034 001314        990-INCREMENT.
       PP 5655-G53 IBM Enterprise COBOL for z/OS and OS/390 3.1.0     CAMRCOB2
       LineID ----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+
  0034 001315            IF  TABLE-SUB LESS THAN 50
  0034 001316                ADD 1 TO TABLE-SUB.
  0034 001317        990-INCREMENT-EX.
  0034 001318            EXIT.
       PP 5655-G53 IBM Enterprise COBOL for z/OS and OS/390 3.1.0     CAMRCOB2
       An "M" preceding a data-name reference indicates that the data-name is mo

        DEFINED   CROSS-REFERENCE OF DATA NAMES   REFERENCES

       000021      ASA-CNTL
       000849      A1                            M1181 1184
       000646      BALANCE-LINE                  M1299
       000640      BALANCE-OUT
       000638      BALANCE-REPORT                M1298
```

By reviewing the code, you determine whether or not it is required. In this case, it is not required for the program and should be removed.

To remove the code in your own programs, you can split the screen again and make the code changes, or wait until you complete the review of the histogram report.

Now return to the current statement in the CAMRCOB2 program.

Action: Type in the command: **cs**

Press Enter.

Result: The application displays the Program Exit Intercept panel.

Action: To continue the advanced demo, enter the command **go options** and press Enter.

or

To end the demo, enter the command **GO**.

Result: If you are continuing the demo, CAMRCOB2 program execution continues with the display of the Advanced Options Menu. Choose another option from the menu.

or

If you are exiting the demo, CAMRCOB2 resumes execution at the normal program exit, and terminates. You return to the Execution Control panel of the Foreground Option. Your previous entries are displayed, letting you begin testing again, or exit. Use PF3 to exit.

Advanced Demo Session for Assembler

Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Initial Intercept for CAMRASM displayed on your screen.

Note: CAMRASM is an ISPF application.

Include Unconditional Breakpoints

Now you are going to set unconditional breakpoints at a number of procedure names in the demo program. In the basic demo you set an unconditional breakpoint using the **u** line command, which is equivalent to the **UNCOND** command. However, a quicker way to include unconditional breakpoints for a test session is to place the breakpoint commands in a data set member, and simply include that member when the Initial Intercept panel displays.

The ASMINCL data set member of CAI.CAMRSAMP contains the application commands that set the breakpoints required for using the Advanced Options Demo for Assembler. Include this data set member as follows:

Action: Enter the following command at the *Initial* Intercept panel:

include asmincl

Press Enter.

Result: The application processes the commands in the asmincl data set, and returns the following message in the upper-right corner of the screen: BREAKPOINT SET.

```
CAMRASM  ---------- CA InterTest Batch *INITIAL* Intercept ----- BREAKPOINT SET
COMMAND ===>                                                  SCROLL ===> CUR
TRACE=> 000005
                       R:D  00000           4          USING LDATA,R13
 *--> 000000 47F0 F074            00074     5          B     START-CAMRASM(,R15
                                              6 *-------------------------------*
                                              7 * Welcome to the CA InterTest   *
                                              8 * Batch Assembler demonstration *
                                              9 * program. You are now at the   *
                                             10 * Initial Breakpoint panel,     *
                                             11 * which is displayed upon entry *
                                             12 * to the first program to be    *
                                             13 * tested. At this point, other  *
                                             14 * breakpoints can be set and    *
                                             15 * control commands can be       *
                                             16 * issued before the program is  *
                                             17 * executed.                     *
                                             18 *-------------------------------*
 *--> 000004 C3C1D4D9C1E2D440                19          DC    CL8'CAMRASM'
 *-->                                        20          DC    CL9'&SYSDATE'
     00000C F0F561F1F661F0F3                  +          DC    CL9'05/16/03'
 *-->                                        21          DC    CL6'&SYSTIME'
     000015 F1F74BF5F340                      +          DC    CL6'17.53'
 *--> 00001B C1C4E5C1D5E3C1C7               22          DC    C'CA INT
```

Access the Demo Session Options Menu

After setting the required breakpoints, access the Demo Session Options Menu by executing the demo program and choosing the Advanced Options Menu from the Welcome panel. Use the following steps:

Action: From the Initial Intercept panel, type **GO** and press Enter to execute the program.

Result: The demo program begins execution and displays the Welcome panel.

Action: Press **PF2** to access the Options Menu for the Advanced Demo Session.

Result: The Advanced Options Preliminary Screen is displayed, reminding you not to access the Options Menu without including the data set member ASMINCL.

Action: Press Enter to continue.

Result: The Demo Session Options Menu displays.

Each of the Options on the menu is independent and can be performed in any order. To use an Option, follow the appropriate section in this guide. Upon completing an Option, you return to the Menu.

Before you start, you should know that:

- To interrupt a loop-press the ATTN or PA1 key. Try the ATTN key first. If that does not interrupt the loop, your testing environment requires you to use the PA1 key. If using PA1, you may need to press RESET first.

Exit the Advanced Options Demo

You can exit the demo from the Options Menu at any time.

Action: Press **PF3**.

Result: The End Demo Session screen displays.

Action: Press Enter.

Result: The CA InterTest Batch Program Breakpoint Intercept panel displays for an unconditional breakpoint at label GOBACK. (This breakpoint was set by a command in the ASMINCL include file.)

Action: Type **GO** and press Enter to execute the program.

Result: The demo program ends, and the CA InterTest Batch Entry panel displays.

Return to the Advanced Options Demo Session

If you exit the CA InterTest Batch Demo, you can return at any time to perform another advanced option. Just repeat the preliminary steps given in the beginning of this chapter. Remember to enter the following command at the Initial Breakpoint screen:

```
include asmincl
```

Option 1: Time-Controlled Execution

This option illustrates how you can slow the execution of your program. The CA InterTest Batch SLOW command executes one statement in your program every few seconds. The exact time interval is specified when you issue the SLOW command. Using the SLOW command together with a keep window is an especially effective testing technique; it allows you to halt the program whenever you see a problem, without having to determine the breakpoint in advance.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the CA InterTest Batch Demo Session Options Menu displayed on your screen.

This option requires you to interrupt the program by pressing the ATTENTION key or the PA1 key. Before proceeding, locate each of these keys on your keyboard. If you need help in finding them, ask a technical support person **before** you begin.

Action: Key in Option **1**.

Press Enter.

Result: CAMRASM displays the screen that describes what occurs in this part of the demo session.

Action: Press Enter.

Result: CA InterTest Batch halts CAMRASM at an unconditional breakpoint set at label PAY_CALC. The statement is highlighted.

Open a keep window that displays the values of YTD and X_AXIS_YTD by entering the commands specified below. Notice how you can use the short form of the keep command (k) and use the TSO command delimiter (normally a semi-colon) to string multiple commands on a single command line.

Action: Key in the following command:

k ytd; k x_axis_ytd

Result: CA InterTest Batch displays a keep window showing the current values of YTD and X_AXIS_YTD.

A sample screen showing the keep window for YTD and X_AXIS_YTD follows:

```
CAMRASM  ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
COMMAND ===>                                                  SCROLL ===> CUR
TRACE=> 000440 000438 000437 000436 000435 000434 000432 000431 000430 000429 0
      ------------------------------------------------------------------------
      0014CE9C AN    001325 02 YTD                          000000000
      0014CE7F AN    001319 02 X_AXIS_YTD
      ------------------------------------------------------------------------
 0001 000414 D208 D66B C9BE 0066B 009BE   438          MVC   TOTALGROSS,ZEROS
U---> 00041A                             440 PAY_CALC DS    0H
 ---> 00041A 1F00                        441          SLR   R0,R0
 ---> 00041C 4000 D102           00102   442          STH   R0,SUB_4
 ---> 000420                             444 PAY_CALC_LOOP DS 0H
 ---> 000420 45E0 C440           00440   445          BAL   R14,POPULATE_GRAPH
 ---> 000424 4810 D102           00102   447          LH    R1,SUB_4
 ---> 000428 4110 1001           00001   448          LA    R1,1(,R1)
 ---> 00042C 4010 D102           00102   449          STH   R1,SUB_4
 ---> 000430 4910 C9CE           009CE   451          CH    R1,NO_OF_HOURS
 ---> 000434 4740 C420           00420   452          BL    PAY_CALC_LOOP
 ---> 000438 45E0 C7EC           007EC   454          BAL   R14,UPDATE_CHEQUE
 ---> 00043C 47F0 C316           00316   456          B     OPTIONS
 ---> 000440                             458 POPULATE_GRAPH DS 0H
 ---> 000440 4820 D102           00102   459          LH    R2,SUB_4
 ---> 000444 4C20 CDF4           00DF4   460          MH    R2,=Y(L'MONTH_ITEM
```

Issue the SLOW command to execute one statement every two seconds. As the program resumes execution, you are able to see the changing values of YTD and X_AXIS_YTD. In this demo, you are going to halt execution when the X_AXIS_YTD value is MAR by pressing the ATTENTION key.

Action: Key in the command: **slow 2**

Press Enter.

Result: CA InterTest Batch initiates a time-controlled execution of CAMRASM, at a rate of one statement every two seconds.

Action: When the value of X_AXIS_YTD in the keep window is **MAR**, press the **ATTN** or **PA1** key. Try the **ATTN** key first.

Result: The application displays the Attention Intercept panel, as shown in the following panel.

A sample Attention Intercept panel follows:

```
CAMRASM  ---------- CA InterTest Batch ATTENTION Intercept ------------------
COMMAND ===>                                                  SCROLL ===> CSR
TRACE=> 000474 000473 000470 000468 000467 000466 000463 000461 000460 000459 0
     ------------------------------------------------------------------------
     0013EE9C AN    001323 02 YTD                          017539242
     0013EE7F AN    001317 02 X_AXIS_YTD                   Mar
     ------------------------------------------------------------------------
                      R:2  00000         462          USING MONTH,R2
 0004 00044C D202 D657 2000 00657 00000   463          MVC   X_AXIS_YTD,MONTH_I
                                          464          DROP  R2
 0003 000452 4820 D102            00102   466          LH    R2,SUB_4
              OPTIONS MENU PROCESSING
       Active Usings: CAMRASM(X'1000'),R12  LDATA(X'1000'),R13
       Loc  Object Code    Addr1 Addr2  Stmt   Source Statement
 0003 000456 4C20 CDF6            00DF6   467          MH    R2,=Y(L'X_AXIS_ITE
 0003 00045A 4122 D65B            0065B   468          LA    R2,X_AXIS_LINE(R2)
                      R:2  00000         469          USING XAXIS,R2
 0003 00045E D201 2000 CA84 00000 00A84   470          MVC   X_AXIS_ITEM,TOKEN_
                                          471          DROP  R2
 0003 000464 4820 D102            00102   473          LH    R2,SUB_4
 0003 000468 4C20 CDF8            00DF8   474          MH    R2,=Y(L'MONTHLY_AM
 0003 00046C 4122 CAAC            00AAC   475          LA    R2,MONTHLY_THIS(R2
                      R:2  00000         476          USING MGROSS,R2
```

Review what you just did.

- You opened a keep window for two variables whose values you wished to view and compare.
- You used the command **slow 2** to have the application slowly execute the program. However, during the program execution, you could monitor the values of the variables in the keep window.
- You used the ATTENTION or PA1 key to stop program execution, and the Attention Intercept panel displayed.

This combination of using keep windows, the slow command, and an Attention Intercept allows you to carefully control and monitor your program execution, while tracking the values of specific variables.

Note: Attention Intercept panel is only displayed when you press the ATTN or PA1 key and then the application executes a statement in a program that is currently being monitored.

Remove the Keep Window

Before returning to the Options Menu of the advanced demo, remove the keep window as follows:

Action: Key in the command: **remove all**
Press Enter.

Result: The application removes the keep window for YTD and X_AXIS_YTD on the Attention Intercept panel.

Continue Execution

Action: Type **GO** and press Enter to continue execution.

Result: CAMRASM resumes execution, and displays the Options Menu.

This concludes Option 1: Time Controlled Execution. Select another option and go to the appropriate section in this guide, or exit the demo using **PF3**.

Option 2: Set Conditional Breakpoints

This option shows you how to set and use conditional breakpoints.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

Action: Key in Option **2**.

Press Enter.

Result: CAMRASM displays the screen that describes what occurs in this part of the demo session:

Action: Press Enter.

Result: The application halts CAMRASM at an unconditional breakpoint. The statement at the breakpoint is highlighted.

Access the screen used to set **conditional** breakpoints. From this screen, you can set both the conditions and any commands you want executed when the specified condition is met.

Action: Key the following in the Command field: **cond**

Press Enter.

Result: The application displays the When panel. WHEN is a synonym for COND.

Action: Make the following entries on the When panel:

Press Enter.

```
---------------------- CA InterTest Batch WHEN PANEL ------------------------
OPTION ===>S
   S - SET A WHEN CONDITION
   R - RESET A WHEN CONDITION
   L - OR BLANK, LIST THE WHEN CONDITIONS

SPECIFY DATA AREA NAME(S) AND COMPARISON CONDITION:
   WHEN-NAME    ===> ytdcond
   DATA-AREA-1  ===> subtotal
   OPERATOR     ===> ge        (EQ, NE, LT, GT, LE, GE, =, <>, <, >, <=, >=)
   DATA-AREA-2  ===> 000060000
   AFTER        ===> N         (Y/N)

COMMANDS TO BE EXECUTED AT WHEN CONDITION:
===>  k subtotal; set totalgross = 000200000; k totalgross
```

Result: The application displays the following message in the top right corner of the When panel: WHEN CONDITION SET.

Action: Press **PF3** to exit the When panel.

Result: The application brings you to the Intercept panel.

Action: Continue execution by typing **GO** and pressing Enter.

Result: CAMRASM resumes execution until the condition you set for the YTDCOND breakpoint is met; when the value of SUBTOTAL is greater than or equal to 60000, the application halts execution, and executes the commands you entered for the conditional breakpoint; it displays a keep window for SUBTOTAL, sets TOTALGROSS to 200000, and then displays TOTALGROSS in the keep window. Your screen should look like the Intercept panel in the following screen.

A sample Intercept panel showing condition YTDCOND has been met follows:

```
CAMRASM  ---------- CA InterTest Batch WHEN YTDCOND BEFORE Intercept ---------
 COMMAND ===>                                                SCROLL ===> CUR
 TRACE=> 000545 000544 000543 000542 000541 000539 000538 000537 000536 000523
       ------------------------------------------------------------------------
       001428CE AN    001329 02 SUBTOTAL                    00009716H
       001428B3 AN    001324 02 TOTALGROSS                   000200000
       ------------------------------------------------------------------------
 GR 0-7 00000000 00000002 1F1A0C90 1F1A0FA6 00142A1E FFEBD5E2 00000000 00000000
    8-F 00000000 00000000 00000000 00000000 9F1A01C8 00142248 9F1A06A2 00000000
       ------------------------------------------------------------------------
  0003 00050E F384 D686 D058 00686 00058   543          UNPK  SUBTOTAL,LDWORK(5)
  0003 000514 96F0 D68E      0068E         544          OI    SUBTOTAL+8,X'F0'
  0002 000518 D208 D67D D686 0067D 00686   545          MVC   BILL_YTD,SUBTOTAL
                                           546          DROP  R2
  0002 00051E 07FE                         548          BR    R14
  ---> 000520                              550 DEMONSTRATE_SPLIT_SCREEN DS 0H
                                           551 *-----------------------------
                                           552 * This section of the demo
                                           553 * shows you how to SPLIT the
                                           554 * screen ..
                                           555 *-----------------------------
  ---> 000520 58F0 D050           00050    556          L     R15,LDLINK@
                                           557          CALL (15),(DSPLY,COBDPN
```

Now remove the conditional breakpoint before continuing.

Action: Return to the When panel by entering the command: **when**

Press Enter.

Result: The When panel displays.

Action: Key in the following on the When panel:

OPTION: ===> r

WHEN NAME ===> ytdcond

Press Enter

Result: The application displays the following message in the top right corner of the When panel: WHEN DELETED.

Action: Press **PF3** to exit the When panel.

Result: The application brings you to the previous Intercept panel.

Remove the Keep Window and Continue Execution

Remove the keep window before continuing execution.

Action: Key in the following command: **remove all**

Press Enter.

Result: The keep window is removed.

Return to the Options Menu by continuing execution.

Action: Type **GO** and press Enter.

Result: CAMRASM resumes execution and displays the Options Menu.

This concludes Option 2: Conditional Breakpoints. Select another option and go to the appropriate section in this guide, or exit the demo using **PF3**.

Option 3: Update Source Code

This option illustrates how you can use the ISPF or split-screen capability from any screen. One way to use this feature is to update your source code as soon as you find errors during your test session.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

Action: Key in Option **3**.

Press Enter.

Result: CAMRASM displays the screen that describes what occurs in this part of the demo session.

Action: Press Enter.

Result: The application halts CAMRASM at an unconditional breakpoint at the label SPLIT_SCREEN.

Action: Tab about halfway down your screen and press the SPLIT PF key defined in your ISPF profile (normally PF2).

Result: The ISPF Primary Option Menu displays in the bottom half of your screen.

Update Your Source Code

If you had found an error in your program code, you can use the ISPF menu to access and edit your source code before continuing with your test session.

For example, in the basic demo, the application stopped execution of the program because it detected an S0C7 ABEND. You were able to continue execution by dynamically changing the value of Tasknum to zero. This was a one-time patch. To correct the problem, you need to update your source code so that Tasknum is properly initialized. You can do this from any Intercept panel by splitting your screen to edit your source code.

Continue Execution

Action: Remove your ISPF split-screen using the command: **=x**

Result: You return to a full-screen display of the unconditional breakpoint shown in the previous panel.

Action: Type **GO** and press Enter to continue execution.

Result: You return to the Demo Options Menu.

This concludes Option 3: Update Source Code. Select another option and go to the appropriate section in this guide, or exit the demo using **PF3**.

Option 4: Interrupt a Looping Program

This option illustrates how to interrupt and verify that you have a looping program. It also illustrates the use of the skip command.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

This option requires you to interrupt the program by pressing the **ATTENTION** key or the **PA1** key. Before proceeding, locate each of these keys on your keyboard. If you need help, ask a technical support person **before** you begin.

Action: Key in Option **4**.

Press Enter

Result: CAMRASM displays the screen that describes what occurs in this part of the demo session.

Action: Press Enter.

Result: The application halts CAMRASM at an unconditional breakpoint set at ORDER_CALC.

Note: If the Frequency Counter arrows and numbers are not displayed along the left side of your screen, turn the frequency display on by entering the command: **freq**.

Action: Continue execution by typing **GO** and pressing enter.

Result: CAMRASM resumes execution.

Notice the system light remains on. There are additional breakpoints and screen panels in the demo program code, yet all you see is the system light. This is symptomatic of a looping program. Halt the program to see what is going on.

Action: Press the **ATTN** or **PA1** key. Try the **ATTN** key first. If that does not work, your testing environment requires you to use the **PA1** key. Press **RESET** before pressing the **PA1** key to unlock the terminal keyboard.

Result: The Attention Intercept panel displays.

Continue with the Continuing from the Attention Routine section.

Note: If you do not successfully stop execution with the ATTN or PA1 key before 1024 executions of the demo program loop, you reach a safety net conditional breakpoint. If you reach this breakpoint, instructions for continuing are given in the Continuing from the "Safety Net" Conditional Breakpoint section.

Continue from the “Safety Net” Conditional Breakpoint

Follow this section only if you see the breakpoint in the following panel. If you do not, continue with the Continuing from the Attention Routine section.

To continue with the looping program demo, you must press the **ATTN** or **PA1 key** before 1024 executions of the loop this time. To continue the program loop, use the following steps:

1. Find label ORDER_CALC: **fp order_calc**
2. Press Enter to go to the label.
3. Type a **g** to the left of the line:

   XC ORDER_SUB_TOTAL,ORDER_SUB_TOTAL

4. Press Enter to continue the loop execution.
5. Press the **ATTN** key or the **RESET**, then **PA1** keys. Try the **ATTN** key first. If that does not work, your testing environment requires you to use the **PA1** key.

   Note: Even if **PA1** stopped the slow execution in option 1, you may need to press **ATTN** to stop a loop.

You should see the Attention Intercept panel.

To return to the Options Menu:

1. Enter the command: **go options**
2. Press Enter.

   You return to the Options Menu. You can try the same option again, even if you did not complete it the first time, or select another option from the menu.

Continue from the Attention Routine

The application now displays the Attention Intercept panel.

CAMRASM keeps executing the TOTAL_ORDER routine. The current statement is highlighted.

Look at the frequency counters to the left of the program statements. They show repeated execution of the TOTAL_ORDER statements, indicating a loop. To investigate the logic in TOTAL_ORDER, you can set a keep window for SUB_6 and then execute slowly using the SLOW command. This time you use the abbreviated form of the keep command: **k**.

Action: Key in the following command to set the keep window: **k sub_6**
Press Enter.

Result: The application displays a keep window for the variable SUB_6.

A sample screen showing the Keep Window to View SUB_6 follows:

```
CAMRASM  ---------- CA InterTest Batch STEP BEFORE Intercept -----------------
OMMAND ===>                                                  SCROLL ===> CUR
TRACE=> 000652 000651 000649 000648 000647 000646 000638 000637 000641 000640 0
      ------------------------------------------------------------------------
      0013F92E HX    001237 02 SUB_6                      0001
      ------------------------------------------------------------------------
                       R:2  00000         650          USING ORDER,R2
 1131 0005CE F873 D058 2002 00058 00002   651          ZAP   LDWORK,UNIT_PRICE
 1130 0005D4 FC73 D058 2006 00058 00006   652          MP    LDWORK,NO_OF_WIDGE
 1130 0005DA D204 D0F3 D05B 000F3 0005B   653          MVC   ITEM_TOTAL,LDWORK+
                                          654          DROP  R2
 1130 0005E0 4810 D106            00106   656          LH    R1,SUB_6
 1130 0005E4 4110 1001            00001   657          LA    R1,1(,R1)
 1130 0005E8 4010 D106            00106   658          STH   R1,SUB_6
 1130 0005EC 4810 D10C            0010C   660          LH    R1,LOOP_OUT
 1130 0005F0 4110 1001            00001   661          LA    R1,1(,R1)
 1130 0005F4 4010 D10C            0010C   662          STH   R1,LOOP_OUT
 1129 0005F8 D201 D106 C9CC 00106 009CC   664          MVC   SUB_6,ONE
 1129 0005FE 07FE                         666          BR    R14

 ---> 000600                              668 DEMONSTRATE_PREV_COMMAND DS 0H
                                          669 *-----------------------------*
                                          670 * This section of the demo      *
```

Action: Key in the following command to execute one CAMRASM statement every two seconds: **slow 2**

Press Enter.

Result: CAMRASM continues a slow execution. As the code executes, you can see the code looping, and you can see the corresponding value of SUB_6 in the keep window.

Notice in the sample screen that SUB_6 changes from 1 to 2, but back to 1 again. SUB_6 is being reset to 1, and causing the program to loop. The value of SUB_6 is causing the TOTAL_ORDER routine to continue to be executed.

Action: Halt execution again by pressing the ATTENTION or PA1 key.

After the Attention Routine message displays, press Enter.

Result: The application brings you to another Attention Intercept panel. The current statement is highlighted.

To dynamically correct the loop-condition, you could use the SKIP command to bypass the offending statement. The **s** line command is another form of the SKIP command.

Action: Tab down to the statement: 'MVC SUB_6,ONE' and key an **s** to the left of the line.

Result: The application skips the statement that sets SUB_6 to 1, and executes the code as if the statement was removed.

Action: Back up to the first statement in the TOTAL_ORDER procedure, 'LH R2,SUB_6', and type a **g** to the left of the line, to continue execution from that point.
Press Enter

Result: CAMRASM resumes execution from the line where you entered the g, and continues execution until it reaches the next unconditional breakpoint, as shown in the following panel.

The following sample screen shows an unconditional breakpoint at ORDER_CALC_EX:

```
CAMRASM  ---------- CA InterTest Batch ATTENTION Intercept -------------------
COMMAND ===>                                                  SCROLL ===> CUR
TRACE=> 000652 000651 000649 000648 000647 000646 000638 000637 000641 000640 0
      ------------------------------------------------------------------------
      0013F92E HX    001237 02 SUB_6                      0001
      ------------------------------------------------------------------------
 1129 0005BA 4740 C5B0           005B0   641          BL    ORDER_CALC_LOOP
U---> 0005BE                             643 ORDER_CALC_EX DS 0H
 ---> 0005BE 47F0 C316           00316   644          B OPTIONS
 1131 0005C2                             646 TOTAL_ORDER DS 0H
 1131 0005C2 4820 D106           00106   647          LH    R2,SUB_6
 1131 0005C6 4C20 CE00           00E00   648          MH    R2,=Y(ORDER_LEN)
 1131 0005CA 4122 D6E4           006E4   649          LA    R2,CUSTOMER_ORDERS
                     R:2  00000          650          USING ORDER,R2
 1131 0005CE F873 D058 2002 00058 00002  651          ZAP   LDWORK,UNIT_PRICE
 1130 0005D4 FC73 D058 2006 00058 00006  652          MP    LDWORK,NO_OF_WIDGE
 1130 0005DA D204 D0F3 D05B 000F3 0005B  653          MVC   ITEM_TOTAL,LDWORK+
                                         654          DROP  R2
 1130 0005E0 4810 D106           00106   656          LH    R1,SUB_6
 1130 0005E4 4110 1001           00001   657          LA    R1,1(,R1)
 1130 0005E8 4010 D106           00106   658          STH   R1,SUB_6
 1130 0005EC 4810 D10C           0010C   660          LH    R1,LOOP_OUT
 1130 0005F0 4110 1001           00001   661          LA    R1,1(,R1)
```

CAMRASM finally exited out of the loop that executed the TOTAL_ORDER routine and then continued until it reached the unconditional breakpoint set at ORDER_CALC_EX (one of the unconditional breakpoints included in the data set member asmincl).

You can see from the keep window that the value of SUB_6 is 5. This is the proper value to permit the exit of the ORDER_CALC routine. Thus, the use of the skip command allowed us to dynamically fix the program code that caused the loop, and continue execution.

Remove the Keep Window and Skip Command

Before going back to the options menu, remove the keep window and skip command as follows.

Action: Key in the command: **remove all**

Tab down to the 'MVC SUB_6,ONE' statement in the TOTAL_ORDER routine and type over the S to the left of the statement with an **x**.

Press Enter.

Result: The keep window is removed, and the skip line command is removed.

Action: Continue execution by typing **GO** and pressing Enter.

Result: CAMRASM resumes execution and displays the Demo Session Options Menu.

This concludes Option 4: Interrupt a Program Loop. Select another option and go to the appropriate section in this guide, or exit the demo using **PF3**.

Option 5: Trace Program Execution

This option illustrates the use of the PREVIOUS and ADVANCE commands to help you trace program execution.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

Action: Key in Option **5**.

Press Enter.

Result: CAMRASM displays the screen that describes what occurs in this part of the demo session.

Action: Press Enter.

Result: The application halts CAMRASM at an unconditional breakpoint that you included at the beginning of the demo.

Suppose you want to trace the execution path of the program prior to this point. You can have the application scroll back and display a previously executed statement using the PREV command. The PREV command lets you view previously executed statements, but does not cause any statements to execute. After you scroll back using the PREV command, you can scroll forward and retrace execution using the ADVANCE command. You may specify a number from 1 to 999 with the PREV and ADVANCE commands to scroll more than one statement at a time.

The PREV and ADVANCE commands do not change the statement where CAMRASM resumes execution; the PREV and ADVANCE only scroll the statements in the trace.

Action: Key in the following in the Command field: **prev 9**

Press Enter.

Result: The application displays the ninth previous statement executed.

Now you can retrace the execution of the nine statements using the ADVANCE command. The short form of the ADVANCE command is A or AD. (You must enter the A on the command line.)

Action: Key in the command: **a**

Press Enter.

Result: The application highlights the next statement executed.

Action: Repeat the Advance step a few more times.

Result: You can follow the execution order of the program by following the highlighted lines as you advance through the program trace. In effect, using PREV and ADVANCE gives you a simulated instant replay of program execution.

Return to the Current Statement

Now you can return to the current statement, that is, the statement where execution was halted prior to the unconditional breakpoint.

Action: Enter the command: **cs**

Press Enter.

Result: The application displays the original Breakpoint Intercept panel shown in the beginning of this option.

Action: Key in the command: **go**

Press Enter

Result: CAMRASM resumes execution and displays the Advanced Options Demo Screen. You can continue with any option or exit.

This concludes Option 5: Trace Program Execution. Select another option and continue with the appropriate section in this guide, or press **PF3** to end the demo.

Option 7: Histogram Report

This option illustrates how to create and view the histogram report of statement execution, which is useful in identifying dead code.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

Action: Key in Option **7**.

Press Enter.

Result: CAMRASM displays the screen that describes what occurs in this part of the demo session.

Action: Press Enter.

Result: The application halts CAMRASM at an unconditional breakpoint set at the program exit.

Prior to exiting the program, you can request a histogram report using the HIST command.

Action: Key in the command: **hist**

Press Enter.

Result: The application returns the following message:

'HISTOGRAM SUCCESSFULLY WRITTEN'

View the Histogram Report

The report is written to the data set associated with the ddname INT1REPT that can be viewed by splitting the screen.

Action: Tab a few lines down your screen and press the SPLIT PF key defined in your ISPF profile (normally PF2).

Result: The ISPF Main Menu displays in the split-screen area.

Action: Select the ISPF Browse Facility and request a display of the data set associated with the ddname INT1REPT. You allocated this data set in Advanced Demo Preliminaries in this appendix. The suggested data set name was: userid.INT1REPT

Result: The first page of the histogram report displays.

Page down to the end of the report using the maximum command with PF8.

Action: Type in the command: **m**

Press PF8.

Result: The last page of the histogram report displays.

Action: Remove your split-screen using the command: **x**

Result: You return to a full-screen display of the Program Exit Intercept.

Use the Find Statement (FS) command to display the code that was never executed.

Action: Type in the command:

fs 902

Press Enter.

Result: The application displays the requested line of code.

Note: Depending on your assemble options, line 902 may not be the label WRITE_ERROR. If this is the case, use the statement number you found by scrolling up from the end of the histogram report.

By reviewing the code, you can determine whether or not it is required. In this case, it is an error routine and should be retained.

If unexecuted code needs to be removed from your own programs, you can split the screen again and make the code changes, or wait until you complete the review of the histogram report.

Now return to the current statement in the CAMRASM program.

Action: Type in the command: **cs**

Press Enter.

Result: The application displays the Program Exit Intercept panel.

Action: To continue the advanced demo, enter the command **go options** and press Enter.

or

To end the demo, enter the command **GO**.

Result: If you are continuing the demo, CAMRASM program execution continues with the display of the Advanced Options Menu. Choose another option from the menu.

or

If you are exiting the demo, CAMRASM resumes execution at the normal program exit, and terminates. You return to the Execution Control panel of the Foreground Option. Your previous entries are displayed, allowing you to begin testing again, or exit. Use PF3 to exit.

Advanced Demo Session for PL/I

Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Initial Intercept for CAMRPLI displayed on your screen.

Note: CAMRPLI is an ISPF application.

Include Unconditional Breakpoints

Now you are going to set unconditional breakpoints at a number of procedure names in the demo program. In the basic demo you set an unconditional breakpoint using the **u** line command, which is equivalent to the **UNCOND** command. However, a quicker way to include unconditional breakpoints for a test session is to place the breakpoint commands in a data set member, and simply include that member when the Initial Intercept panel displays.

The PLIINCL data set member of CAI.CAMRSAMP contains the commands that set the breakpoints required for using the Advanced Options Demo for PL/I. Include this data set member as follows:

Action: Enter the following command at the *Initial* Intercept panel:

include pliincl

Press Enter.

Result: The application processes the commands in the pliincl data set, and returns the following message in the upper-right corner of the screen: BREAKPOINT SET.

```
CAMRPLI  ---------- CA InterTest Batch *INITIAL* Intercept ----- BREAKPOINT SET
COMMAND ===>                                                    SCROLL ===> CUR
TRACE=> 000001
      ************************** Top of Data ********************************
                          SOURCE LISTING
          STMT LEV NT
 *-->
              1      0  CAMRPLI: PROC OPTIONS (MAIN);


/*------------------------------------------------*/
                          /* Welcome to the CA InterTest Batch    */
                          /* PL/I demonstration program. You are now at the
*/
                          /* Initial Breakpoint panel which is displayed
*/
                          /* upon entry to the first program to be tested.
*/
                          /* At this point, other breakpoints can be set
*/
                          /* and control commands can be issued before the
*/
                          /* program is executed.
*/

/*------------------------------------------------*/
              2   1  0     DCL  REPORT                    FILE RECORD
SEQUENTIA
                                                         ENV (F,
RECSIZE(80));
              3   1  0     DCL  1 LINE_OUT,
                                   3 ASA_CNTL            CHAR(1),
```

Access the Demo Session Options Menu

After setting the required breakpoints, you can access the Demo Session Options Menu by executing the demo program and choosing the Advanced Options Menu from the Welcome panel. Use the following steps:

Action: From the Initial Intercept panel, type **GO** and press Enter to execute the program.

Result: The demo program begins execution and displays the Welcome panel.

Action: Press **PF2** to access the Options Menu for the Advanced Demo Session.

Result: The Advanced Options Preliminary Screen displays, reminding you not to access the Options Menu without including the data set member PLIINCL.

Action: Press Enter to continue.

Result: The Demo Session Options Menu displays.

A sample Demo Session Options Menu follows:

Each of the options on the menu is independent and can be performed in any order. To use an option, follow the appropriate section in this guide. Upon completing an option, you return to the Menu.

Before you start, you should know that:

To interrupt a loop-press the ATTN or PA1 key. Try the ATTN key first. If that does not interrupt the loop, your testing environment requires you to use the PA1 key. If using PA1, you may need to press RESET first.

Exit the Advanced Options Demo

You can exit the demo from the Options Menu at any time.

Action: Press **PF3**.

Result: The End Demo Session screen displays.

Action: Press Enter.

Result: The Program Breakpoint Intercept panel displays for an unconditional breakpoint at label RETURN_TO_OPSYS. (This breakpoint was set by a command in the PLIINCL include file.)

Action: Type **GO** and press Enter to execute the program.

Result: The demo program ends, and the Entry panel displays.

Return to the Advanced Options Demo Session

If you exit the demo, you can return at any time to perform another advanced option. Just repeat the preliminary steps given in the beginning of this chapter. Remember to enter the following command at the Initial Breakpoint screen:

```
include pliincl
```

Option 1: Time-Controlled Execution

This option illustrates how you can slow the execution of your program. The SLOW command executes one statement in your program every few seconds. The exact time interval is specified when you issue the SLOW command. Using the SLOW command together with a keep window is an especially effective testing technique; it lets you halt the program whenever you see a problem, without having to determine the breakpoint in advance.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

This option requires you to interrupt the program by pressing the ATTENTION key or the PA1 key. Before proceeding, locate each of these keys on your keyboard. If you need help in finding them, ask a technical support person **before** you begin.

Action: Key in Option **1**.
Press Enter.

Result: CAMRPLI displays the screen that describes what occurs in this part of the demo session.

Action: Press Enter.

Result: The application halts CAMRPLI at an unconditional breakpoint set at procedure PAY_CALC. The procedure statement is highlighted.

Open a keep window that displays the values of YTD and X_AXIS_YTD by entering the commands specified below. Notice how you can use the short form of the keep command (k) and use the TSO command delimiter (normally a semi-colon) to string multiple commands on a single command line.

Action: Key in the following command:
k ytd; k x_axis_ytd

Result: The application displays a keep window showing the current values of YTD and X_AXIS_YTD.

A sample screen showing the keep window for YTD and X_AXIS_YTD follows:

```
CAMRPLI  ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
COMMAND ===>                                                  SCROLL ===> CUR
TRACE=> 000168 000094 000093 000092 000091 000088 000087 000086 000085 000084
0

------------------------------------------------------------------------------
      1F1A84EF NP-S  000090 02 YTD                          +0000000.00
      1F1A84F7 AN    000089 03 X_AXIS_YTD

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

U--->      168   1  0     PAY_CALC: PROC ( I );

           169   2  0        DCL I                        PIC '999';
           170   2  0        DCL J                        FIXED BIN(15);


 --->      171   2  0        DO J = 1 TO I BY 1;
 --->      172   2  1           CALL POPULATE_GRAPH ( J );
 --->      173   2  1           END;

                          /* CHEQUE_LINE                 = ' ';             */
 --->      174   2  0        CHEQUE_AMOUNT               = TOTALGROSS;
 --->      175   2  0        RETURN;

 --->      176   2  0     END PAY_CALC;
```

Issue the SLOW command to execute one statement every two seconds. As the program resumes execution, you are able to see the changing values of YTD and X_AXIS_YTD. In this demo, you are going to halt execution when the X_AXIS_YTD value is MAR by pressing the ATTENTION key.

Action: Key in the command: **slow 2**

Press Enter.

Result: The application initiates a time-controlled execution of CAMRPLI, at a rate of one statement every two seconds.

Action: When the value of X_AXIS_YTD in the keep window is **MAR**, press the **ATTN** or **PA1** key. Try the **ATTN** key first.

Result: The Attention Intercept panel displays, as shown in the following panel.

A sample Attention Intercept panel follows:

```
 CAMRPLI  ---------- CA InterTest Batch ATTENTION Intercept ------------------
COMMAND ===>                                                  SCROLL ===> CUR
TRACE=> 000185 000184 000183 000182 000177 000172 000173 000186 000185 000184
0

-------------------------------------------------------------------------------
      1F1A84EF NP-S  000090 02 YTD                          +0256273.52
      1F1A84F7 AN    000089 03 X_AXIS_YTD                   MAR

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

 0003      182   2  0       X_AXIS_YTD                = MONTH_ITEM ( I );
 0003      183   2  0       X_AXIS_R ( I )            = TOKEN_ITEM;
 0003      184   2  0       YTD                      = YTD + MONTHLY_AMOUNT
 0002      185   2  0       TOTALGROSS                = YTD;
 0002      186   2  0       RETURN;

 --->      187   2  0    END POPULATE_GRAPH;

 --->      188   1  0    BILL_CALC: PROC;

           189   2  0       DCL   I                      FIXED BIN(15);
           190   2  0       DCL 1 BILLS_BILLS_BILLS,
                                3 BILLS_MONTHLY,
                                  5 BILLING_AMOUNT (12) FIXED DEC(7,2)
INIT
```

Review what you just did.

- You opened a keep window for two variables whose values you wished to view and compare.
- You used the command **slow 2** to have the application slowly execute the program. However, during the program execution, you could monitor the values of the variables in the keep window.
- You used the ATTENTION or PA1 key to stop program execution, and the Attention Intercept panel displays.

This combination of using keep windows, the slow command, and an Attention Intercept lets you carefully control and monitor your program execution, while tracking the values of specific variables.

Note: The Attention Intercept panel is only displayed when the user presses the ATTN or PA1 key and then the application executes a statement in a program that is currently being monitored.

Remove the Keep Window

Before returning to the Options Menu of the advanced demo, remove the keep window as follows:

Action: Key in the command: **remove all**

Press Enter.

Result: The application removes the keep window for YTD and X_AXIS_YTD on the Attention Intercept panel.

Continue Execution

Action: Type **GO** and press Enter to continue execution.

Result: CAMRPLI resumes execution, and displays the Options Menu.

This concludes Option 1: Time Controlled Execution. Select another option and go to the appropriate section in this guide, or exit the demo using **PF3**.

Option 2: Set Conditional Breakpoints

This option shows you how to set and use conditional breakpoints.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

Action: Key in Option **2**.

Press Enter.

Result: CAMRPLI displays the screen that describes what occurs in this part of the demo session.

Action: Press Enter.

Result: The application halts CAMRPLI at an unconditional breakpoint. The statement at the breakpoint is highlighted.

Access the screen used to set **conditional** breakpoints. From this screen, you can set both the conditions and any commands you want executed when the specified condition is met.

Action: Key the following in the Command field: **cond**

Press Enter.

Result: The When panel displays. WHEN is a synonym for COND.

Action: Make the following entries on the When panel:

Press Enter.

```
---------------------- CA InterTest Batch WHEN PANEL ------------------------
OPTION ===>S

   S - SET A WHEN CONDITION
 R - RESET A WHEN CONDITION
   L - OR BLANK, LIST THE WHEN CONDITIONS

SPECIFY DATA AREA NAME(S) AND COMPARISON CONDITION:
   WHEN-NAME    ===> ytdcond
   DATA-AREA-1  ===> subtotal
   OPERATOR     ===> ge        (EQ, NE, LT, GT, LE, GE, =, <>, <, >, <=, >=)
   DATA-AREA-2  ===> 600.00
   AFTER        ===> N         (Y/N)

COMMANDS TO BE EXECUTED AT WHEN CONDITION:
===>  k subtotal; set totalgross = 2000; k totalgross
```

Result: The application displays the following message in the top right corner of the When panel: WHEN CONDITION SET.

Action: Press **PF3** to exit the When panel.

Result: The application brings you to the previous Breakpoint Intercept panel.

Action: Continue execution by typing **GO** and pressing Enter.

Result: CAMRPLI resumes execution until the condition you set for the YTDCOND breakpoint is met; when the value of SUBTOTAL is greater than or equal to 600, CA InterTest Batch halts execution, and executes the commands you entered for the conditional breakpoint; it displays a keep window for SUBTOTAL, sets TOTALGROSS to 2000, and then displays TOTALGROSS in the keep window. Your screen should look like the Intercept panel in the following screen.

A sample intercept panel showing condition YTDCOND has been met follows:

```
 CAMRPLI  ---------- CA InterTest Batch WHEN YTDCOND BEFORE Intercept ---------
COMMAND ===>                                                  SCROLL ===> CUR
TRACE=> 000193 000192 000194 000193 000192 000194 000193 000192 000191 000188
0

------------------------------------------------------------------------------
      1F1A84E6 NP-S  000101 02 SUBTOTAL                    +00971.68
      1F1A84EA NP-S  000090 02 TOTALGROSS                  +0002000.00

------------------------------------------------------------------------------
 0003      192   2  1          SUBTOTAL                 = SUBTOTAL + BILLING_A
 0002      193   2  1          BILL_YTD                  = SUBTOTAL;
 0002      194   2  1        END;
 --->      195   2  0        RETURN;

 --->      196   2  0     END BILL_CALC;

 --->      197   1  0     INITIALIZE_TABLE: PROC;

           198   2  0        DCL (I, J)                   FIXED BIN(15);

 --->      199   2  0        DO I = 1 TO 2;
 --->      200   2  1          DO J = 1 TO 9;
 --->      201   2  2            STATE_NUMBER ( I )      = I;
      5688-235 IBM PL/I for MVS & VM          CAMRPLI: PROC OPTIONS (MAIN);
          STMT LEV NT
 --->      202   2  2            COUNTY_NUM1 ( I, J )    = I;
```

Now remove the conditional breakpoint before continuing.

Action: Return to the When panel by entering the command: **when**

Press Enter.

Result: The When panel displays.

Action: Key in the following on the When panel:

Press Enter

```
----------------------- CA InterTest Batch WHEN PANEL ------------------------
OPTION ===>R

   S - SET A WHEN CONDITION
   R - RESET A WHEN CONDITION
   L - OR BLANK, LIST THE WHEN CONDITIONS

SPECIFY DATA AREA NAME(S) AND COMPARISON CONDITION:
   WHEN-NAME    ===> ytdcond
   DATA-AREA-1  ===>
   OPERATOR     ===>          (EQ, NE, LT, GT, LE, GE, =, <>, <, >, <=, >=)
   DATA-AREA-2  ===>
   AFTER        ===> N        (Y/N)

COMMANDS TO BE EXECUTED AT WHEN CONDITION:
===>
```

Result: The application displays the following message in the top right corner of the When panel: WHEN DELETED.

Action: Press **PF3** to exit the When panel.

Result: The application brings you to the previous Intercept panel.

Remove the Keep Window and Continue Execution

Remove the keep window before continuing execution.

Action: Key in the following command: **remove all**

Press Enter.

Result: The keep window is removed.

Return to the Options Menu by continuing execution.

Action: Type **GO** and press Enter.

Result: CAMRPLI resumes execution and displays the Options Menu.

This concludes Option 2: Conditional Breakpoints. Select another option and go to the appropriate section in this guide, or exit the demo using **PF3**.

Option 3: Update Source Code

This option illustrates how you can use the ISPF split-screen capability from any screen. One way to use this feature is to update your source code as soon as you find errors during your test session.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

Action: Key in Option **3**.

Press Enter.

Result: CAMRPLI displays the screen that describes what occurs in this part of the demo session.

Action: Press Enter.

Result: The application halts CAMRPLI at an unconditional breakpoint at the label SPLIT_SCREEN.

Action: Tab about halfway down your screen and press the SPLIT PF key defined in your ISPF profile (normally PF2).

Result: The ISPF Primary Option Menu displays in the bottom half of your screen.

A sample showing using a split-screen from an Intercept panel follows:

```
CAMRPLI  ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
COMMAND ===>                                                 SCROLL ===> CUR
TRACE=> 000109 000108 000107 000106 000105 000078 000077 000076 000075 000074
0

U--->      109   1  0     SPLIT_SCREEN:

                            GO TO OPTIONS;

                          /*-----------------------------------------------*/
                          /* FROM THIS UNCONDITIONAL BREAKPOINT YOU CAN     */
                          /* PRESS THE PF KEY ASSIGNED TO THE SPLIT COMMAND */
                          /* WHICH TAKES YOU TO THE ISPF MAIN MENU.         */
                          /*-----------------------------------------------*/
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.
                                  CA
                           Islandia Data Center
                                                                       USER01
OPTION ===>                                  CCCCCCCCCCCCCCCCCCCCCCCC CATSO
                                           CCCCCCCCCCCCCCCCCCCCCCCC
06/05/02
                                          CCCC                         14:09
   C  CA Products                        CCCC                         XAD1
   D  SVC Dump Index                     CCCC                 AAAAAAA
   G  Color Graphics Terminal Products   CCCC                AAAAAAAAA
   P  Program Development Facility       CCCC             AAAA   AAAA
```

Update Your Source Code

If you had found an error in your program code, you can use ISPF to access and edit your source code before continuing with your test session.

For example, in the basic demo, the application stopped execution of the program because it detected an S0C7 ABEND. You were able to continue execution by dynamically changing the value of Tasknum to zero. This was a one-time patch. To correct the problem, you need to update your source code so that Tasknum is properly initialized. You can do this from any Intercept panel by splitting your screen to edit your source code.

Continue Execution

Action: Remove your ISPF split-screen using the command: **=x**

Result: You return to a full-screen display of the unconditional breakpoint shown in the previous panel.

Action: Type **GO** and press Enter to continue execution.

Result: You return to the Demo Options Menu.

This concludes Option 3: Update Source Code. Select another option and go to the appropriate section in this guide, or exit the demo using **PF3**.

Option 4: Interrupt a Looping Program

This option illustrates how to interrupt and verify that you have a looping program. It also illustrates the use of the skip command.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

This option requires you to interrupt the program by pressing the **ATTENTION** key or the **PA1** key. Before proceeding, locate each of these keys on your keyboard. If you need help, ask a technical support person **before** you begin.

Action: Key in Option **4**.

Press Enter

Result: CAMRPLI displays the screen that describes what occurs in this part of the demo session.

Action: Press Enter.

Result: The application halts CAMRPLI at an unconditional breakpoint set at ORDER_CALC.

A sample screen showing CAMRPLI stopped at ORDER_CALC follows:

```
 CAMRPLI  ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
COMMAND ===>                                                  SCROLL ===> CUR
TRACE=> 000134 000133 000132 000131 000130 000079 000078 000077 000076 000075
0

U--->      134   1  0     ORDER_CALC:

                           /*-------------------------------------------------*/
                           /* THIS SECTION OF THE DEMO SHOWS YOU HOW TO      */
                           /* INTERRUPT A LOOPING PROGRAM AND CHANGE THE     */
                           /* EXECUTION LOGIC WITHOUT RE-COMPILATION.        */
                           /*-------------------------------------------------*/

                              LOOP_OUT                    = 0;

 --->      135   1  0        CALL TOTAL_ORDER ( );

 --->      136   1  0        GO TO OPTIONS;

 --->      137   1  0     DEMONSTRATE_HIST_COMMAND:

                           /*-------------------------------------------------*/
                           /* THIS SECTION OF THE DEMO SHOWS YOU HOW TO      */
                           /* ISSUE THE HISTOGRAM COMMAND.                   */
                           /*-------------------------------------------------*/
```

Note: If the Frequency Counter arrows and numbers are not displayed along the left side of your screen, turn the frequency display on by entering the command: **freq**.

Action: Continue execution by typing **GO** and pressing Enter.

Result: CAMRPLI resumes execution.

Notice the system light remains on. There are additional breakpoints and screen panels in the demo program code, yet all you see is the system light. This is symptomatic of a looping program. Halt the program to see what is going on.

Action: Press the **ATTN** or **PA1** key. Try the **ATTN** key first. If that does not work, your testing environment requires you to use the **PA1** key. Press **RESET** before pressing the **PA1** key to unlock the terminal keyboard.

Result: The Attention Intercept panel displays.

Continue with the Continuing from the Attention Routine section.

Note: If you do not successfully stop execution with the ATTN or PA1 key before 20,000 executions of the demo program loop, you reach a safety net conditional breakpoint. If you reach this breakpoint, instructions for continuing are given in the Continuing from the "Safety Net" Conditional Breakpoint section.

Continue from the "Safety Net" Conditional Breakpoint

Follow this section only if you see the breakpoint in the following panel. If you do not, continue with the Continuing from the Attention Routine section.

A sample screen showing a Safety Net Conditional Breakpoint of a Demo Loop follows:

```
CAMRPLI  ---------- CA InterTest Batch WHEN LOOPCOND BEFORE Intercept --------
COMMAND ===>                                                  SCROLL ===> CUR
TRACE=> 000219 000218 000217 000216 000221 000220 000219 000218 000217 000216 0
      ------------------------------------------------------------------------
      227A93FA NP-S  000210    ITEM_TOTAL                  +0000969.00
      227A941A NP-S  000213 04 UNIT_PRICE                  +00080.75
                               I  (+2)
      227A941E NP-S  000213 04 NO_OF_WIDGETS               +0000012.
                               I  (+2)
      ------------------------------------------------------------------------
 020K      217   2  1          I                          = I + 1;
 020K      218   2  1          LOOP_OUT                   = LOOP_OUT + 1;
 020K      219   2  1          ITEM_TOTAL                 = UNIT_PRICE ( I ) *
                                                        NO_OF_WIDGETS ( I );
      5688-235 IBM PL/I for MVS & VM         CAMRPLI: PROC OPTIONS (MAIN);
          STMT LEV NT
 020K      220   2  1          I                          = 1;
 020K      221   2  1          ORDER_SUB_TOTAL            = ORDER_SUB_TOTAL + IT
           222   2  1        END;

U--->      223   2  0     TOTAL_ORDER_EX:

                            RETURN;
```

To continue with the looping program demo, you must press the **ATTN** or **PA1 key** before 20,000 executions of the loop this time. To continue the program loop, use the following steps:

1. Type **GO** and press Enter to continue the loop execution.
2. Press the **ATTN** key or the **RESET**, then **PA1** keys. Try the **ATTN** key first. If that does not work, your testing environment requires you to use the **PA1** key.

Note: Even if **PA1** stopped the slow execution in option 1, you may need to press **ATTN** to stop a loop.

You should see the Attention Intercept panel.

To return to the Options Menu:

1. Enter the command: **go options_menu**

   Note: This compiler is an optimizing compiler. Therefore, an abend is possible using the GO <label> command. If this happens, you can return to the Options Menu by restarting the demo application.

2. Press Enter.

   You return to the Options Menu. You can try the same option again, even if you did not complete it the first time, or select another option from the menu.

Continue from the Attention Routine

The application now displays the Attention Intercept panel. A sample showing an Attention Intercept in procedure TOTAL_ORDER follows:

```
CAMRPLI  ---------- CA InterTest Batch ATTENTION Intercept -------------------
COMMAND ===>                                                  SCROLL ===> CUR
TRACE=> 000221 000220 000219 000218 000217 000216 000221 000220 000219 000218 0
 013K      220   2  1          I                          = 1;
 013K      221   2  1          ORDER_SUB_TOTAL            = ORDER_SUB_TOTAL + IT
           222   2  1        END;

U--->      223   2  0     TOTAL_ORDER_EX:

                             RETURN;

 --->      224   2  0     END TOTAL_ORDER;

 --->      225   1  0     PROPAGATE: PROC;

           226   2  0        DCL I                         FIXED BIN(15)   INIT(
           227   2  0        DCL TABLE_SUB                 FIXED BIN(15)   INIT(
           228   2  0        DCL 1 TABLE_OF_SORTS,
                                  3 TABLE_ELEMENT (60)    CHAR(1);

 --->      229   2  0        DO I = 1 TO 17;
 --->      230   2  1          TABLE_SUB                 = INCREMENT ( TABLE_SU
 --->      231   2  1          TABLE_ELEMENT ( TABLE_SUB ) = '+';
 --->      232   2  1          TABLE_SUB                 = INCREMENT ( TABLE_SU
```

CAMRPLI keeps executing the TOTAL_ORDER procedure, as you can see from your Attention Intercept screen display. The current statement is highlighted.

Look at the frequency counters to the left of the program statements. They show repeated execution of the TOTAL_ORDER statements, indicating a loop. To investigate the logic in TOTAL_ORDER, you can set a keep window for I and then execute slowly using the SLOW command. This time you use the abbreviated form of the keep command: **k**.

Action: Key in the following command to set the keep window: **k i**

Press Enter.

Result: The application displays a keep window for the variable I.

A sample screen showing the keep window to View I follows:

```
CAMRPLI  ---------- CA InterTest Batch ATTENTION Intercept ------------------
COMMAND ===>                                                 SCROLL ===> CUR
TRACE=> 000221 000220 000219 000218 000217 000216 000221 000220 000219 000218 0
     ------------------------------------------------------------------------
     1F1A93F2 NB-S  000209    I                            +1.
     ------------------------------------------------------------------------
 013K      220   2  1          I                          = 1;
 013K      221   2  1          ORDER_SUB_TOTAL            = ORDER_SUB_TOTAL + IT
           222   2  1        END;

U--->      223   2  0     TOTAL_ORDER_EX:

                             RETURN;

 --->      224   2  0     END TOTAL_ORDER;

 --->      225   1  0     PROPAGATE: PROC;

           226   2  0        DCL I                         FIXED BIN(15)   INIT(
           227   2  0        DCL TABLE_SUB                 FIXED BIN(15)   INIT(
           228   2  0        DCL 1 TABLE_OF_SORTS,
                                  3 TABLE_ELEMENT (60)    CHAR(1);

 --->      229   2  0        DO I = 1 TO 17;
```

Action: Key in the following command to execute one CAMRPLI statement every two seconds: **slow 2**

Press Enter.

Result: CAMRPLI continues a slow execution. As the code executes, you can see the code looping, and you can see the corresponding value of I in the keep window.

The following sample screen of a SLOW 2 Execution shows a Loop and I problem:

```
CAMRPLI  ---------- CA InterTest Batch STEP BEFORE Intercept -----------------
COMMAND ===>                                                  SCROLL ===> CUR
TRACE=> 000220 000219 000218 000217 000216 000221 000220 000219 000218 000217 0
     ------------------------------------------------------------------------
     1F1A93F2 NB-S  000209    I                          +2.
     ------------------------------------------------------------------------

 013K      216   2  0        DO UNTIL (I = ORDER_ITEMS);
 013K      217   2  1          I                         = I + 1;
 013K      218   2  1          LOOP_OUT                  = LOOP_OUT + 1;
 013K      219   2  1          ITEM_TOTAL                = UNIT_PRICE ( I ) *
                                                        NO_OF_WIDGETS ( I );
     5688-235 IBM PL/I for MVS & VM          CAMRPLI: PROC OPTIONS (MAIN);
         STMT LEV NT
 013K      220   2  1          I                         = 1;
 013K      221   2  1          ORDER_SUB_TOTAL           = ORDER_SUB_TOTAL + IT
           222   2  1        END;

U--->      223   2  0     TOTAL_ORDER_EX:

                             RETURN;

 --->      224   2  0     END TOTAL_ORDER;
```

Notice that I changes from 1 to 2, but back to 1 again. I is being reset to 1, and causing the program to loop. The value of I is causing the TOTAL_ORDER procedure to continue to be executed.

Action: Halt execution again by pressing the ATTENTION or PA1 key.

After the Attention Routine message displays, press Enter.

Result: The application brings you to another Attention Intercept panel. The current statement is highlighted.

To dynamically correct the loop-condition, you could use the SKIP command to bypass the offending statement. The **s** line command is another form of the SKIP command.

Action: Tab down to the statement: I = 1, and key an **s** to the left of the line.

The following sample screen shows using skip to bypass the cause of the loop:

```
CAMRPLI  ---------- CA InterTest Batch ATTENTION Intercept ------------------
COMMAND ===>                                                  SCROLL ===> CSR
TRACE=> 000219 000218 000217 000216 000221 000220 000219 000218 000217 000216 0
      ------------------------------------------------------------------------
      1F1A93F2 NB-S  000209    I                          +2.
      ------------------------------------------------------------------------

 013K      216   2  0        DO UNTIL (I = ORDER_ITEMS);
 013K      217   2  1          I                          = I + 1;
 013K      218   2  1          LOOP_OUT                   = LOOP_OUT + 1;
 013K      219   2  1          ITEM_TOTAL                 = UNIT_PRICE ( I ) *
                                                        NO_OF_WIDGETS ( I );
      5688-235 IBM PL/I for MVS & VM          CAMRPLI: PROC OPTIONS (MAIN);
          STMT LEV NT
S013K     220   2  1          I                          = 1;
 013K      221   2  1          ORDER_SUB_TOTAL            = ORDER_SUB_TOTAL + IT
           222   2  1        END;

U--->      223   2  0     TOTAL_ORDER_EX:

                             RETURN;

 --->      224   2  0     END TOTAL_ORDER;
```

Result: The application skips the statement that sets I to 1, and executes the code as if the statement was removed.

Action: Tab up to the second statement in the DO loop, I = I + 1, and type a **g** to the left of the line, to continue execution from that point

Your entry should match the following panel. The following sample screen shows a continuing execution from the add statement:

```
CAMRPLI  ---------- CA InterTest Batch NEXT BEFORE Intercept --- BREAKPOINT SET
COMMAND ===>                                                  SCROLL ===> CSR
TRACE=> 000219 000218 000217 000216 000221 000220 000219 000218 000217 000216 0
     ------------------------------------------------------------------------
     1F1A93F2 NB-S  000209    I                            +2.
     ------------------------------------------------------------------------

 013K      216   2  0        DO UNTIL (I = ORDER_ITEMS);
g013K      217   2  1          I                          = I + 1;
 013K      218   2  1          LOOP_OUT                   = LOOP_OUT + 1;
 013K      219   2  1          ITEM_TOTAL                 = UNIT_PRICE ( I ) *
                                                        NO_OF_WIDGETS ( I );
     5688-235 IBM PL/I for MVS & VM          CAMRPLI: PROC OPTIONS (MAIN);
         STMT LEV NT
S013K      220   2  1          I                          = 1;
 013K      221   2  1          ORDER_SUB_TOTAL            = ORDER_SUB_TOTAL + IT
           222   2  1        END;

U--->      223   2  0     TOTAL_ORDER_EX:

                             RETURN;

 --->      224   2  0     END TOTAL_ORDER;
```

Press Enter.

Result: CAMRPLI resumes execution from the line where you entered the g, and continues execution until it reaches the next unconditional breakpoint, as shown in the following panel.

The following sample screen shows an unconditional breakpoint at TOTAL_ORDER_EX:

```
CAMRPLI  ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
COMMAND ===>                                                  SCROLL ===> CUR
TRACE=> 000223 000221 000219 000218 000217 000216 000221 000219 000218 000217 0
     ------------------------------------------------------------------------
     1F1A93F2 NB-S  000209    I                            +5.
     ------------------------------------------------------------------------

 013K      216   2  0        DO UNTIL (I = ORDER_ITEMS);
 013K      217   2  1          I                          = I + 1;
 013K      218   2  1          LOOP_OUT                   = LOOP_OUT + 1;
 013K      219   2  1          ITEM_TOTAL                 = UNIT_PRICE ( I ) *
                                                        NO_OF_WIDGETS ( I );
     5688-235 IBM PL/I for MVS & VM          CAMRPLI: PROC OPTIONS (MAIN);
         STMT LEV NT
S013K      220   2  1          I                          = 1;
 013K      221   2  1          ORDER_SUB_TOTAL            = ORDER_SUB_TOTAL + IT
           222   2  1        END;

U--->      223   2  0     TOTAL_ORDER_EX:

                             RETURN;

 --->      224   2  0     END TOTAL_ORDER;
```

CAMRPLI finally exited out of the DO loop and continued until it reached the unconditional breakpoint set at TOTAL_ORDER_EX (one of the unconditional breakpoints included in the data set member PLIINCL).

You can see from the keep window that the value of I is 5. This is the proper value to permit the exit of the DO loop. Thus, the use of the skip command allowed us to dynamically fix the program code that caused the loop, and continue execution.

Remove the Keep Window and Skip Command

Before going back to the options menu, remove the keep window and skip command as follows.

Action: Key in the command: **remove all**

Tab down to the I = 1 statement and type over the S to the left of the statement with an **x**.

Press Enter.

Result: The keep window is removed, and the skip line command is removed.

Action: Continue execution by typing **GO** and pressing Enter.

Result: CAMRPLI resumes execution and displays the Demo Session Options Menu.

This concludes Option 4: Interrupt a Program Loop. Select another option and go to the appropriate section in this guide, or exit the demo using **PF3**.

Option 5: Trace Program Execution

This option illustrates the use of the PREVIOUS and ADVANCE commands to help you trace program execution.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

Action: Key in Option **5**.

Press Enter.

Result: CAMRPLI displays the screen that describes what occurs in this part of the demo session.

The following sample screen shows Option 5: Trace Program Execution:

```
*************************************************************************
*****                                                               *****
*****             CA InterTest Batch Demo Session                   *****
*****                 Tracing program execution                     *****
*****                                                               *****
*************************************************************************

  CA InterTest Batch gives you the ability to trace the execution of
  your program up to the point where a breakpoint occurs.
  Here is what will happen:

  1. CA InterTest Batch halts the DEMO program at an unconditional
     breakpoint.
  2. You use a combination of the Prev and Advance commands to
     scroll through the program statements in execution order.
  3. You issue the CS command to re-display the Current Statement
     to be executed.


     Press ENTER to continue or PF3 to return to the Options Menu.
```

Action: Press Enter.

Result: The application halts CAMRPLI at an unconditional breakpoint that you included at the abeginning of the demo.

The following sample screen shows CAMRPLI at an unconditional breakpoint:

```
CAMRPLI  ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
COMMAND ===>                                                 SCROLL ===> CUR
TRACE=> 000168 000118 000117 000116 000115 000114 000113 000112 000111 000110
0

U0001      168   1  0     PAY_CALC: PROC ( I );

           169   2  0        DCL I                        PIC '999';
           170   2  0        DCL J                        FIXED BIN(15);


 0001      171   2  0        DO J = 1 TO I BY 1;
 0005      172   2  1           CALL POPULATE_GRAPH ( J );
 0005      173   2  1           END;

                         /* CHEQUE_LINE                = ' ';            */
 0001      174   2  0        CHEQUE_AMOUNT              = TOTALGROSS;
 0001      175   2  0        RETURN;

 --->      176   2  0     END PAY_CALC;

 0005      177   1  0     POPULATE_GRAPH: PROC ( I );

           178   2  0        DCL   I                      FIXED BIN(15);
           179   2  0        DCL 1 MONTH_THING            STATIC,
```

Suppose you want to trace the execution path of the program prior to this point. You can have the application scroll back and display a previously executed statement using the PREV command. The PREV command lets you view previously executed statements, but does not cause any statements to execute. After you scroll back using the PREV command, you can scroll forward and retrace execution using the ADVANCE command. You may specify a number from 1 to 999 with the PREV and ADVANCE commands to scroll more than one statement at a time.

The PREV and ADVANCE commands do not change the statement where CAMRPLI resumes execution; the PREV and ADVANCE only scroll the statements in the trace.

Action: Key in the following in the Command field: **prev 9**

Press Enter.

Result: The application displays the ninth previous statement executed.

The following screen shows the ninth previous statement:

```
CAMRPLI  ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
COMMAND ===>                                                 SCROLL ===> CUR
TRACE=> 000168 000118 000117 000116 000115 000114 000113 000112 000111 000110
0

 0001      110   1  0     DEMONSTRATE_PREV_COMMAND:

                            CALL ISPLINK ( DSPLY,
                                           COBDPN75 );

                           /*-------------------------------------------------*/
                           /* THIS SECTION OF THE DEMO SHOWS YOU              */
                           /* THE BENEFITS OF USING THE PREV                  */
                           /* AND ADVANCE COMMANDS.                           */
                           /*-------------------------------------------------*/

 0001      111   1  0        COBDPN71_LENGTHS            = 4;

 0001      112   1  0        CALL ISPLINK ( VCOPY,
                                           VCOPY_COBDPN71,
                                           VCOPY_COBDPN71_LENGTHS,
                                           VCOPY_COBDPN71_VALUES,
                                           VCOPY_MOVE );

 0001      113   1  0        IF VCOPY_COBDPN71_EIBAID    = 'PF03' THEN
```

Now you can retrace the execution of the nine statements using the ADVANCE command. The short form of the ADVANCE command is A or AD. (You must enter the A on the command line.)

Action: Key in the command: **a**

Press Enter.

Result: The application highlights the next statement executed.

Action: Repeat the Advance step a few more times.

The following screen shows advancing through the program trace:

```
CAMRPLI  ---------- CA InterTest Batch UNCOND BEFORE Intercept --------------
COMMAND ===>                                                 SCROLL ===> CUR
TRACE=> 000168 000118 000117 000116 000115 000114 000113 000112 000111 000110
0

 0001      110   1  0     DEMONSTRATE_PREV_COMMAND:

                            CALL ISPLINK ( DSPLY,
                                           COBDPN75 );

                          /*-----------------------------------------------*/
                          /* THIS SECTION OF THE DEMO SHOWS YOU            */
                          /* THE BENEFITS OF USING THE PREV                */
                          /* AND ADVANCE COMMANDS.                         */
                          /*-----------------------------------------------*/

 0001      111   1  0        COBDPN71_LENGTHS           = 4;

 0001      112   1  0        CALL ISPLINK ( VCOPY,
                                           VCOPY_COBDPN71,
                                           VCOPY_COBDPN71_LENGTHS,
                                           VCOPY_COBDPN71_VALUES,
                                           VCOPY_MOVE );

 0001      113   1  0        IF VCOPY_COBDPN71_EIBAID    = 'PF03' THEN
```

Result: You can follow the execution order of the program by following the highlighted lines as you advance through the program trace. In effect, using PREV and ADVANCE gives you a simulated instant replay of program execution.

Option 5: Trace Program Execution

This option illustrates the use of the PREVIOUS and ADVANCE commands to help you trace program execution.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

Action: Key in Option **5**.

Press Enter.

Result: CAMRPLI displays the screen that describes what occurs in this part of the demo session.

Action: Press Enter.

Result: The application halts CAMRPLI at an unconditional breakpoint that you included at the abeginning of the demo.

The following sample screen shows CAMRPLI at an unconditional breakpoint:

```
CAMRPLI  ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
COMMAND ===>                                                  SCROLL ===> CUR
TRACE=> 000168 000118 000117 000116 000115 000114 000113 000112 000111 000110
0

U0001      168   1  0     PAY_CALC: PROC ( I );

           169   2  0        DCL I                         PIC '999';
           170   2  0        DCL J                         FIXED BIN(15);


 0001      171   2  0        DO J = 1 TO I BY 1;
 0005      172   2  1           CALL POPULATE_GRAPH ( J );
 0005      173   2  1           END;

                            /* CHEQUE_LINE                = ' ';             */
 0001      174   2  0           CHEQUE_AMOUNT              = TOTALGROSS;
 0001      175   2  0           RETURN;

 --->      176   2  0     END PAY_CALC;

 0005      177   1  0     POPULATE_GRAPH: PROC ( I );

           178   2  0        DCL   I                       FIXED BIN(15);
           179   2  0        DCL 1 MONTH_THING             STATIC,
```

Suppose you want to trace the execution path of the program prior to this point. You can have the application scroll back and display a previously executed statement using the PREV command. The PREV command lets you view previously executed statements, but does not cause any statements to execute. After you scroll back using the PREV command, you can scroll forward and retrace execution using the ADVANCE command. You may specify a number from 1 to 999 with the PREV and ADVANCE commands to scroll more than one statement at a time.

The PREV and ADVANCE commands do not change the statement where CAMRPLI resumes execution; the PREV and ADVANCE only scroll the statements in the trace.

Action: Key in the following in the Command field: **prev 9**

Press Enter.

Result: The application displays the ninth previous statement executed.

The following screen shows the ninth previous statement:

```
CAMRPLI  ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
COMMAND ===>                                                 SCROLL ===> CUR
TRACE=> 000168 000118 000117 000116 000115 000114 000113 000112 000111 000110
0

 0001      110   1  0     DEMONSTRATE_PREV_COMMAND:

                            CALL ISPLINK ( DSPLY,
                                           COBDPN75 );

                           /*-----------------------------------------------*/
                           /* THIS SECTION OF THE DEMO SHOWS YOU            */
                           /* THE BENEFITS OF USING THE PREV                */
                           /* AND ADVANCE COMMANDS.                         */
                           /*-----------------------------------------------*/

 0001      111   1  0        COBDPN71_LENGTHS           = 4;

 0001      112   1  0        CALL ISPLINK ( VCOPY,
                                           VCOPY_COBDPN71,
                                           VCOPY_COBDPN71_LENGTHS,
                                           VCOPY_COBDPN71_VALUES,
                                           VCOPY_MOVE );

 0001      113   1  0        IF VCOPY_COBDPN71_EIBAID    = 'PF03' THEN
```

Now you can retrace the execution of the nine statements using the ADVANCE command. The short form of the ADVANCE command is A or AD. (You must enter the A on the command line.)

Action: Key in the command: **a**
Press Enter.

Result: The application highlights the next statement executed.

Action: Repeat the Advance step a few more times.

The following screen shows advancing through the program trace:

```
CAMRPLI  ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
COMMAND ===>                                                  SCROLL ===> CUR
TRACE=> 000168 000118 000117 000116 000115 000114 000113 000112 000111 000110
0

 0001      110   1  0     DEMONSTRATE_PREV_COMMAND:

                            CALL ISPLINK ( DSPLY,
                                           COBDPN75 );

                           /*-----------------------------------------------*/
                           /* THIS SECTION OF THE DEMO SHOWS YOU            */
                           /* THE BENEFITS OF USING THE PREV                */
                           /* AND ADVANCE COMMANDS.                         */
                           /*-----------------------------------------------*/

 0001      111   1  0        COBDPN71_LENGTHS           = 4;

 0001      112   1  0        CALL ISPLINK ( VCOPY,
                                           VCOPY_COBDPN71,
                                           VCOPY_COBDPN71_LENGTHS,
                                           VCOPY_COBDPN71_VALUES,
                                           VCOPY_MOVE );

 0001      113   1  0        IF VCOPY_COBDPN71_EIBAID    = 'PF03' THEN
```

Result: You can follow the execution order of the program by following the highlighted lines as you advance through the program trace. In effect, using PREV and ADVANCE gives you a simulated instant replay of program execution.

Return to the Current Statement

Now you can return to the current statement, that is, the statement where execution was halted prior to the unconditional breakpoint.

Action: Enter the command: **cs**

Press Enter.

Result: The application displays the original Breakpoint Intercept panel shown in the beginning of this option.

Action: Key in the command: **go**

Press Enter

Result: CAMRPLI resumes execution and displays the Advanced Options Demo screen. You can continue with any option or exit.

This concludes Option 5: Trace Program Execution. Select another option and continue with the appropriate section in this guide, or press **PF3** to end the demo.

Option 6: Work with Indexed Table Items

This section of the demo session shows you a fast and easy way to inspect and modify the contents of indexed data items. It also demonstrates how to use the STEP command to single-step execution.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

Action: Select Option **6**.

Press Enter.

Result: CAMRPLI displays a screen that describes what occurs in this part of the demo session.

Action: Press Enter.

Result: CAMRPLI resumes execution.

The next screen you see is an Abend Intercept panel. The application has automatically halted CAMRPLI because it detected a data exception in the highlighted statement.

The following screen shows CAMRPLI at an S0C7 Abend Intercept:

```
CAMRPLI  ---------- CA InterTest Batch ABEND S0C7 Intercept ------------------
COMMAND ===>                                                SCROLL ===> CUR
TRACE=> 000128 000127 000126 000125 000206 000205 000204 000203 000202 000201
0

 0001      128   1  0       DISTRICT ( A1, B1, C1 )     =
                               DISTRICT ( A1, B1, C1 ) + 1;
          ------------------------------------------------------------
          |       THE FOLLOWING AUTOMATIC BREAKPOINT OCCURRED:         |
 --->     |   INTERCEPT IN PROGRAM CAMRPLI  AT #000128 REASON: ABEND S0C7 |
          |       PRESS ENTER TO REMOVE THIS MESSAGE.                  |
          ------------------------------------------------------------
                         /* THIS SECTION OF THE DEMO SHOWS YOU HOW TO      */
                         /* INSPECT AND CHANGE THE VALUE OF SUBSCRIPTED    */
                         /* TABLE ITEMS.                                   */
                         /*------------------------------------------------*/

 0001      130   1  0     DEMONSTRATE_LOOP_DETECTION:

                         /*------------------------------------------------*/
                         /* THIS SECTION OF THE DEMO SHOWS YOU HOW TO      */
                         /* INTERRUPT A LOOPING PROGRAM AND CHANGE THE     */
                         /* EXECUTION LOGIC  _ WITHOUT RE_COMPILATION.     */
                         /*------------------------------------------------*/
```

If you recall, this is the same type of error detected and corrected in the basic demo session. To continue execution, you need to find the data item causing the S0C7 ABEND and dynamically modify it.

Looking at the highlighted breakpoint statement, you might suspect the variable is at fault; it probably was not initialized correctly. The variable in this case is an indexed table entry.

Action: Press Enter to remove the ABEND message box.

Result: The ABEND message box is removed.

Display an Indexed Table Entry

You can quickly inspect the value of the **DISTRICT (A1, B1,** C1) field directly from this screen by entering a **k** to the left of the highlighted statement.

Action: Tab to the highlighted statement and type **k** (for keep).

Press Enter.

Result: The application displays the current value of DISTRICT. It also displays the value of each subscript item, because the k line command requests a display of each item on that line.

The following panel displays a Keep Window for the Table Entry:

```
CAMRPLI  ---------- CA InterTest Batch ABEND S0C7 Intercept ------------------
COMMAND ===>                                                  SCROLL ===> CUR
TRACE=> 000128 000127 000126 000125 000206 000205 000204 000203 000202 000201
0

------------------------------------------------------------------------------
      1F1A8531 NP-S  000037 04 DISTRICT                    ?000.
                                A1  (+1)
                                B1  (+3)
                                C1  (+5)

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

 0001      128   1  0        DISTRICT ( A1, B1, C1 )     =
                                DISTRICT ( A1, B1, C1 ) + 1;

                           /* TASK_STRUCTURE.TASK_TEXT    = ' ';            */
 --->      129   1  0        GO TO OPTIONS;

                           /*-----------------------------------------------*/
                           /* THIS SECTION OF THE DEMO SHOWS YOU HOW TO     */
                           /* INSPECT AND CHANGE THE VALUE OF SUBSCRIPTED   */
                           /* TABLE ITEMS.                                  */
                           /*-----------------------------------------------*/

 0001      130   1  0     DEMONSTRATE_LOOP_DETECTION:
```

The question mark that precedes the value of DISTRICT in the keep window tells us this field does not contain a valid value. This is the error that triggered the automatic breakpoint.

The keep window also tells us which table entry is at fault. The values of each subscript item in this window indicate the invalid data occurs at
element (1, 3, 5).

Dynamically Correct an Uninitialized Table Item

To dynamically correct this error you need to modify main storage so the field contains valid data. You can use the SET command to set the value to zero. The application sets the variable using the proper format.

Action: Type the command:

set district (a1,b1,c1) = zero

Press Enter.

The following panel corrects the uninitialized Table Item:

```
CAMRPLI  ---------- CA InterTest Batch ABEND S0C7 Intercept -----------------
COMMAND ===> set district (a1,b1,c1) = zero                 SCROLL ===> CUR
TRACE=> 000128 000127 000126 000125 000206 000205 000204 000203 000202 000201
0

------------------------------------------------------------------------------
      1F1A8531 NP-S  000037 04 DISTRICT                    ?000.
                                A1  (+1)
                                B1  (+3)
                                C1  (+5)

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

 0001      128   1  0        DISTRICT ( A1, B1, C1 )     =
                                DISTRICT ( A1, B1, C1 ) + 1;

                           /* TASK_STRUCTURE.TASK_TEXT    = ' ';            */
 --->      129   1  0        GO TO OPTIONS;

                           /*-----------------------------------------------*/
                           /* THIS SECTION OF THE DEMO SHOWS YOU HOW TO     */
                           /* INSPECT AND CHANGE THE VALUE OF SUBSCRIPTED   */
                           /* TABLE ITEMS.                                  */
                           /*-----------------------------------------------*/

 0001      130   1  0     DEMONSTRATE_LOOP_DETECTION:
```

Result: Thea application dynamically modifies the value in main storage for DISTRICT (A1, B1, C1), and displays the new value in the keep window. The message on the top right of the panel reads: SET COMPLETE.

Single-Step Execution

Use the step command to execute just one line of code. The step command lets you determine how many lines are executed when you use the go command. For example, to execute 1 line at a time, or single-step, use the command: **step 1**.

Action: Type the command: **step 1**

Press Enter.

Result: The application sets the step count to 1 and displays the following message: **STEP COUNT SET**.

The following panel shows setting the step count:

```
CAMRPLI  ---------- CA InterTest Batch ABEND S0C7 Intercept ---- STEP COUNT SET
COMMAND ===>                                                   SCROLL ===> CUR
TRACE=> 000128 000127 000126 000125 000206 000205 000204 000203 000202 000201
0

------------------------------------------------------------------------------
      1F1A8531 NP-S  000037 04 DISTRICT                    +000.
                                A1  (+1)
                                B1  (+3)
                                C1  (+5)

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

 0001      128   1  0        DISTRICT ( A1, B1, C1 )     =
                                DISTRICT ( A1, B1, C1 ) + 1;

                           /* TASK_STRUCTURE.TASK_TEXT    = ' ';            */
 --->      129   1  0        GO TO OPTIONS;

                           /*-----------------------------------------------*/
                           /* THIS SECTION OF THE DEMO SHOWS YOU HOW TO     */
                           /* INSPECT AND CHANGE THE VALUE OF SUBSCRIPTED   */
                           /* TABLE ITEMS.                                  */
                           /*-----------------------------------------------*/

 0001      130   1  0     DEMONSTRATE_LOOP_DETECTION:
```

Action: Execute CAMRPLI by entering the command: **GO** and pressing Enter.

Result: The statement that increments the DISTRICT table item is executed, and then the application halts execution and displays a Step Count Intercept panel. Statement 129 is now the current line, and is highlighted.

Remove the Step Count and Keep Window

When you want to end single-stepping, reset the step count to zero. This turns single stepping off. Remove the step count and keep window before continuing execution.

Action: Type in the command:

step 0; remove all

Press Enter.

Result: The application resets the step count and removes the keep window.

To display the current status of the step count, enter the step command without any count number as follows:

Action: Type in the command: **step**

Press Enter.

Result: The application displays the value of the step count in the top-right corner; in this case, the message is: NO STEP COUNT SET.

Action: Resume execution of CAMRPLI by entering **go**.

Result: CAMRPLI resumes execution, and you return to the Options Menu.

This concludes Option 6: Work with Indexed Table Items. Select another option and continue with the appropriate section in this guide, or press **PF3** to end the demo.

Option 7: Histogram Report

This option illustrates how to create and view the histogram report of statement execution, which is useful in identifying dead code.

Note: Before proceeding, be sure you have completed the steps outlined in Advanced Demo Preliminaries in this appendix and you have the Demo Session Options Menu displayed on your screen.

Action: Key in Option **7**.

Press Enter.

Result: CAMRPLI displays the screen that describes what occurs in this part of the demo session.

Action: Press Enter.

Result: The application halts CAMRPLI at an unconditional breakpoint set at the program exit.

The following screen shows the breakpoint intercept at label RETURN_TO_OPSYS:

```
CAMRPLI  ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
COMMAND ===>                                                  SCROLL ===> CUR
TRACE=> 000058 000142 000235 000234 000233 000239 000237 000232 000231 000239
0

U--->       58   1  0     RETURN_TO_OPSYS:

                             RELEASE ISPLINK;
 --->       59   1  0        CLOSE FILE ( REPORT );
 --->       60   1  0        RETCODE = DM_RC;
 --->       61   1  0        CALL PLIRETC ( RETCODE );
 --->       62   1  0        RETURN;

 0001       63   1  0     OPTIONS_MENU:

                            /*---------------------------------------------*/
                            /* INITIALIZE THE LENGTHS IN CASE OF RECURSION.   */
                            /*---------------------------------------------*/


      5688-235 IBM PL/I for MVS & VM          CAMRPLI: PROC OPTIONS (MAIN);
          STMT LEV NT
                            CALL ISPLINK ( DSPLY,
                                           PLIDPN70 );

 0001       64   1  0        OPTION_I                   = 1;
```

Prior to exiting the program, you can request a histogram report using the HIST command.

Action: Key in the command: **hist**

Press Enter.

Result: The application returns the following message:

'HISTOGRAM SUCCESSFULLY WRITTEN'

View the Histogram Report

The report is written to the data set associated with the ddname INT1REPT that can be viewed by splitting the screen.

Action: Tab a few lines down your screen and press the SPLIT PF key defined in your ISPF profile (normally PF2).

Result: The ISPF Main Menu displays in the split-screen area.

Action: Select the ISPF Browse Facility and request a display of the data set associated with the ddname INT1REPT. You allocated this data set in the Advanced Demo Preliminaries section. The suggested data set name was: userid.INT1REPT

Note: In the following example, the report is viewed in SDSF.

Result: The first page of the histogram report displays. A sample report is shown in the following panel.

The following screen shows Browsing the Histogram DD in SDSF:

```
CAMRPLI  ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
 COMMAND ===>                                                 SCROLL ===> CSR
 TRACE=> 000058 000142 000235 000234 000233 000239 000237 000232 000231 000239

 U--->       58   1  0     RETURN_TO_OPSYS:
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 SYSVIEW 11.5 CA11 ----------------- Output ----------------- 05/02/14
18:01:17
 Command ====>                                                Scroll *===>
CSR
 ------------------------------------------------ Lvl 4 Row 1-35 Col
1-80/115
 USER001  TSU   27743  DDname - INT1REPT  Stepname - CATSO     Procstep -
CATSO

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

...+....10...+....20...+....30...+....40...+....50...+....60...+....70...+....
CA InterTest/Batch       Execution Histogram For Program-ID: CAMRPLI
Date:

000001 0001 |*
000042 0001 |*
000043 0001 |*
000044 0001 |*
000045 0001 |*
000046 0001 |*
000047 0001 |*
000048 0001 |*
000049 0001 |*
000050 ---- |
000051 ---- |
```

Page down to the end of the report using the maximum command with PF8

Action: Type in the command: **m**

Press PF8.

Result: The last page of the histogram report displays, as shown in the following panel.

The following screen shows the last page of the histogram report:

```
CAMRPLI  ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
 COMMAND ===>                                                  SCROLL ===> CSR
 TRACE=> 000058 000142 000235 000234 000233 000239 000237 000232 000231 000239

 U--->       58   1  0     RETURN_TO_OPSYS:
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 SYSVIEW 11.5 CA11 ----------------- Output ----------------- 05/02/14 18:06:44
 Command ====>                                                  Scroll *===>CSR
 GSVX006I End of data ---------------------- Lvl 4 Row 165-195/195 Col1-80/115
 RYARO02  TSU   27743  DDname - INT1REPT  Stepname - CATSO     Procstep -CATSO

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

...+....10...+....20...+....30...+....40...+....50...+....60...+....70...+....
000234 0017 |*
000235 0001 |*
000236 ---- |
000237 0034 |*
000239 0034 |*
000240 ---- |
000241 ---- |
000242 ---- |
            ----+----|----+----|----+----|----+----|----+----|----+----|----+-
            *NOTE* ASTERISKS REPRESENT A SCALE OF 00000130 EXECUTION(S) PERAST
CA InterTest/Batch       Execution Histogram For Program-ID: CAMRPLI     Date:
Session Number: 54   For Userid RYARO02
Number of executable statements: 174
Number of statements executed during this test: 147
Percentage of statements executed during this test: 84
```

In the previous panel, you can see that a number of lines were never executed. Check to see if these lines in the program are dead code and should be removed.

Note: In our example, there is unexecuted code starting at line 240. Note the corresponding line number in your program that is unexecuted.

Action: Remove your split-screen using the command: **x**

Result: You return to a full-screen display of the Program Exit Intercept shown in the beginning of this option.

Use the Find Statement (FS) command to display the code that was never executed.

Action: Type in the command:

fs 240

Press Enter.

Result: The application displays the requested line of code.

The following screen displays the unexecuted code:

```
CAMRPLI  ---------- CA InterTest Batch UNCOND BEFORE Intercept ---------------
COMMAND ===>                                                  SCROLL ===> CUR
TRACE=> 000058 000142 000235 000234 000233 000239 000237 000232 000231 000239 0
                              RETURN ( I + 1 );
 --->      240   2  0        ELSE
                              RETURN ( I );

 --->      241   2  0     END INCREMENT;

 --->      242   1  0  END CAMRPLI;

      5688-235 IBM PL/I for MVS & VM         CAMRPLI: PROC OPTIONS (MAIN);
                                                          ATTRIBUTE AND CROS
      DCL NO.    IDENTIFIER                    ATTRIBUTES AND REFERENCES

      3          ASA_CNTL                      /* IN LINE_OUT */ AUTOMATIC UN
      5          A1                            AUTOMATIC ALIGNED BINARY FIXED
                                               125,128,128
      18         BALANCE_LINE                  /* IN BALANCE_OUT */ AUTOMATIC
      18         BALANCE_OUT                   AUTOMATIC /* STRUCTURE */
      17         BALANCE_REPORT                AUTOMATIC /* STRUCTURE */
      188        BILL_CALC                     ENTRY RETURNS(DECIMAL /* SINGL
                                               102
      101        BILL_YTD                      /* IN BILLING_VALUES */ AUTOMA
```

By reviewing the code, you can determine whether or not it is required. In this case, it is required for the program and should be retained.

If unexecuted code needs to be removed from your own programs, you can split the screen again and make the code changes, or wait until you complete the review of the histogram report.

Now return to the current statement in the CAMRPLI program.

Action: Type in the command: **cs**

Press Enter.

Result: The application displays the Program Exit Intercept panel.

Action: To continue the advanced demo, enter the command **go options** and press Enter.

or

To end the demo, enter the **GO** command.

Result: If you are continuing the demo, CAMRPLI program execution continues with the display of the Advanced Options Menu. Choose another option from the menu.

or

If you are exiting the demo, CAMRPLI resumes execution at the normal program exit, and terminates. You return to the Execution Control panel of the Foreground Option. Your previous entries are displayed, allowing you to begin testing again, or exit. Use PF3 to exit.

Appendix D: DDnames and Sample CLISTs

This section contains the following topics:

DDnames Required for Proper Execution

This section contains a list of the ddnames used by the application and a brief description of their use.

DCB information, where given, is that which is set and used by the application.

DDname	Description
INT1LOAD	Defines the partitioned data set (PDS) that contains the load modules. This library is built during installation.
INT1MSGL	Defines the partitioned data set that contains the application messages. This library is built during installation.
INT1PARM	Defines the partitioned data set that contains the application parameters. This library is built during installation. **Note:** The parameters are internal to the application and are documented. For information on how these parameters are documented, see Step 15. Customize Options in the chapter "Installation" in the *Installation Guide*.
INT1PNLL	Defines the partitioned data set that contains the panel definitions. This library is built during installation.
INT1PROF	Defines the partitioned data set used to maintain user-related data from session to session. This library must be defined by the installation. For information on how to allocate the profile library, see Step 14a. Allocate PROFLIB of the "Installation" chapter in the *Installation Guide*.

Optional DDnames

These ddnames are used during certain processes. If you are using the designated application facility, you must allocate these files before using that part of your product.

It is usually prudent to include the necessary allocation of ddnames to DSN in the CLIST that invokes the application.

DDname	Description
INT1PRNT *	Defines a sequential file used to print the screen images. If it is not allocated, the screen image feature is disabled. For more information on printing screens, see the print key description in Program Function and Program Access Keys in the chapter "System Overview." **Note:** This DD is used only under CA Roscoe. ■ LRECL=121, BLKSIZE=6171, RECFM=FBA
INT1REPT	Defines a sequential file used to store the HISTogram and XSUM reports. If it is not allocated, the HIST and XSUM commands are disabled. For more information on the reports, see the HIST and XSUM Report Command descriptions in the chapter "Debugging Commands." ■ LRECL=131,BLKSIZE=6157,RECFM=FBA
INT1ALIB	Defines the partitioned data set that contains allocation sets saved under option 3, ALLOCATION. If it is not allocated, the save and retrieve features are disabled. For more information on allocating data sets and saving the allocations, see the chapter "Allocations Facility for ISPF." ■ LRECL=150, BLKSIZE=3150, RECFM=FB
INT1CLIB	Defines the partitioned data set that contains application commands to be used by the INCLUDE command. If it is not allocated, the INCLUDE command is disabled. For more information, see the Include Control Command in the chapter "Debugging Commands." ■ LRECL=80, RECFM=FB
INT1CLOG	Defines a sequential file used for the application session log. If it is not allocated, logging is disabled. For more information on the session log, see Session Log Facility (Review Debugging Session) in the chapter "Debugging Commands." ■ LRECL=80, BLKSIZE=6160, RECFM=FB
SYSOUT	Although SYSOUT is not used by the application, it is recommended that it be allocated to the terminal to prevent ABENDS while processing COBOL DISPLAY or EXHIBIT verbs that may have been left in the program.

Sample CLISTs to Invoke the Application

The CAMRCLIB contains five sample CLISTs for invoking the application in different program testing environments. The following table describes the sample CLISTs:

Name	Description
MR91CLST	Sample CLIST for application testing.
MR91CIMS	Sample CLIST for testing of IMS programs in foreground.
MR91CBTS	Sample CLIST for testing of IMS programs using BTS input in foreground.
MR91CDB2	Sample CLIST for testing of DB2 programs in foreground.
CAMRDRVR	Sample CLIST that provides a front-end to the other four CLISTs.

Other CLISTs

The application contains several other CLISTs for your use. They are described in the following table:

Name	Description
MR91STRT	This gets called under an ISPF NEWAPPL and calls the program to start the application. MR91KEYS can be called from this CLIST to set the PF Keys
MR91KEYS	This gets called from the CLIST MR91STRT to set up initial PF key settings for this APPL. By default, all of the PF keys are set to an initial value. By default, the PF keys will only be set once. To have the PF keys reset, modify the variable CHANGED.
INT1ALIB	This CLIST conditionally creates an ALIB data set. It can be called from your startup CLIST.
INT1CLIB	This CLIST conditionally creates a command library data set. It can be called from your startup CLIST.
INT1CLOG	This CLIST conditionally creates a session log data set. It can be called from your startup CLIST.
INT1PROF	This CLIST conditionally creates an application profile data set. It can be called from your startup CLIST.
INT1REPT	This CLIST conditionally creates a user report data set. It can be called from your startup CLIST.

Appendix E: Messages and Diagnostics

This section contains the following topics:

CA InterTest Batch Messages

This section contains application messages and descriptions, where appropriate. The messages are listed in alphanumeric order, by message number.

ABA00 PROGRAM CHECK INTERRUPTION WHILE PROCESSING I/O REQUEST

ABA03 WHEN TASK ATTEMPTED TO TERMINATE, ONE OR MORE SUBTASKS NOT TERMINED

ABA05 AREA TO BE FREED BY FREEMAIN MACRO OVERLAPPED

Explanation: Free area in virtual storage or part of the area was still fixed in real storage.

ABA06 PRIOR REQUEST FOR QUEUED MODULE NOW REQUESTED BY LINK, LOAD, ATT OR XCTL MACRO

ABA0A R-TYPE FREEMAIN MACRO SPECIFIED AREA THAT OVERLAPPED

Explanation: Free area in virtual storage or part of area was still fixed in real storage.

ABA13 DATA SET REQUESTED BY OPEN MACRO NOT FOUND (TAPE ONLY)

ABA14 CLOSE MACRO ENCOUNTERED I/O ERROR WHILE ATTEMPTING PARTIAL RELEASE OF SPACE IN DASD

CAMRA011 NO MEMBERS IN LIBRARY

Explanation: The library specified contains no members.

CAMRA012 NO MATCHES

Explanation: The pattern specified produced no matches.

CAMRA013 CAN'T OPEN LIBRARY

Explanation: An error occurred when opening specified library.

CAMRA014 FREED

Explanation: All possible ddnames were freed.

CAMRA028 MEMBER IS EMPTY

Explanation: The member does not contain any JCL.

CAMRA029 NOT AN EXEC STATEMENT

Explanation: Selected statement is not a valid JCL statement.

CAMRA033 CANNOT LOAD JCLCHECK

Explanation: Load module JCLCHECK cannot be found or loaded.

CAMRA034 NO PROCLIB

Explanation: You must supply at least one PROCLIB

CAMRA035 INVALID DSN

Explanation: The data set name is not valid.

CAMRA036 DDNAME NOT ALLOCATED

Explanation: The ddname specified was not pre-allocated as required.

CAMRA037 SELECTION NOT ALLOWED

Explanation: Member selection list is not allowed; member name is required.

CAMRA038 SOME ALLOCATIONS FAILED

Explanation: Review for highlighted allocations errors.

CAMRA090 DDNAME REQUIRED

Explanation: Data set cannot be allocated without ddname.

CAMRA092 BROWSE/EDIT ERROR

Explanation: ISPF is unable to browse or edit selected data set.

CAMRA093 INVALID LINE COMMAND

Explanation: Enter ALLOCATE, BROWSE, CONVERT (CLIST), EDIT, or FREE.

CAMRA094 INVALID LIBRARY TYPE

Explanation: Only PDS, CA Librarian, and CA Panvalet are supported.

CAMRA095 NOTHING TO ALLOCATE/FREE

Explanation: The member contains only messages or empty lines.

CAMRA097 NO ALLOS TO ALLOCATE

Explanation: ALIB member is empty or contains only message allos.

CAMRA098 INTERNAL ERROR - PLIST

Explanation: Invalid PLIST supplied.

CAMRA099 CANNOT LOAD FAIR/OPEN ERROR

Explanation: Cannot load CA Librarian FAIR modules or OPEN failed.

CAMRA101 INVALID LINE CMD

Explanation: Enter BROWSE, EDIT/SELECT, ALLOC, FREE, or CONVERT.

CAMRA102 MEMBER NAME REQUIRED

Explanation: Enter a member name in the provided field.

CAMRA103 MEMBER REPLACED

Explanation: Member was replaced in data set.

CAMRA104 INVALID LINE CMD

Explanation: Enter F to free, B to browse, or E to edit.

CAMRA105 INVALID LINE CMD

Explanation: Enter B to browse or S to select for JCL conversion.

CAMRA107 INVALID LINE CMD

Explanation: Enter I, D, R, S, AL, or F line command.

CAMRA301 OPEN ERROR

Explanation: OPEN failed for ALIB library.

CAMRA302 MEMBER NOT FOUND

Explanation: Requested member was not found in library.

CAMRA303 FIND ERROR

Explanation: FIND caused an error other than not found.

CAMRA304 INVALID RECORDS

Explanation: Invalid ALLO records found in member.

CAMRA305 EMPTY MEMBER

Explanation: Member requested contains no records.

CAMRA306 NEW MEMBER

Explanation: Member not found in library; new member created.

CAMR910E SYM RECORD WITHOUT SYMBOLIC OPERAND

CAMR911E SYM REPLACEMENT RECORD NOT FOUND

CAMR912E “PGM=” NOT FOUND IN PROGRAM RECORD

CAMR914E IBM DB2 “SYSTSIN” DD NOT FOUND

CAMR915E IBM DB2 “LIB” STATEMENT ERROR

CAMR916E IBM DB2 “LIB” STMT INVALID DSNAME

CAMR917E IBM DB2 “PROGRAM” STATEMENT ERROR

CAMR950I SVC INTERCEPT INSTALLED SUCCESSFULLY

CAMR951E SVC NUMBER WAS NOT SPECIFIED

CAMR952E INVALID ARGUMENT LENGTH

CAMR954E VALUE OUT OF RANGE (MUST BE 200-255)

CAMR955E UNABLE TO LOAD SVC INTERCEPT MODULE

CAMR956E UNABLE TO LOAD SVC INTERCEPT PROLOGUE

CAMR957E UNABLE TO INSTALL SVC INTERCEPT

CAMR960E SYSTEM NOT INITIALIZED FOR BATCH LINK, REPLY “RETRY” OR “CANCEL”

CAMR961E INT1OPTS DD MISSING OR INVALID

Explanation: The INT1OPTS DD statement was not included in the JCL for a Batch Link step, or the file referenced by the DD statement is the wrong format for Batch Link options. Correct the JCL and resubmit the job.

CAMR962E EXEC= OPTION MISSING OR INVALID

Explanation: The INT1OPTS DD statement did not contain any EXEC= option, or the name specified was invalid. Correct the JCL and resubmit the job.

CAMRS01W BLSCHTBL NOT ANCHORED.

Explanation: Batch Link Schedule table not created and anchored off the CAAT. Review the initialization job log to determine cause. Rerun the initialization after correcting the cause.

CAMRS02E INT1SKUT DD NOT FOUND.

Explanation: The DD statement with INT1SKUT name was not found in the job stream. Correct error and rerun the appropriate job.

CAMRS03E CANNOT DETERMINE DSORG. EXPORT ABORTED.

Explanation: Export operation has aborted because the DSORG of the dataset pointed to by INT1SKUT DD statement is unknown. DSORG must be either PS or PO.

CAMRS04E INVALID MEMBER NAME SPECIFIED FOR SEQUENTIAL DATASET. EXPORT ABORTED.

Explanation: Export operation has aborted because a member name was specified for a sequential dataset. Correct the error and rerun Export.

CAMRS05E PDS WITH NO MEMBER NAME SPECIFIED.

Explanation: Schedule found a partitioned dataset but there was no member name specified. Correct the error and rerun the job.

CAMRS06I SCHEDULE EXPORTED TO ==> xxxxx

Explanation: Schedule is being exported to an external dataset named xxxxx.

CAMRS07I TOTAL RECORDS EXPORTED TO INT1SKUT DD: nnnn

Explanation: During a Schedule Export operation, nnnn records were exported to the INT1SKUT DD.

CAMRS08I BLSCHTBL IS EMPTY.

Explanation: Batch Link Schedule Table was found anchoring off CAAT, however it does not contain any active schedules.

CAMRS09E DSORG NOT SUPPORTED, EXPORT ABORTED.

Explanation: Schedule Export operation has aborted due to the DSORG of the dataset pointed to by INT1SKUT DD statement is of a type not supported. DSORG must be either PS or PO.

CAMRS10I INT1SKUT DD OPENED.

Explanation: Informational message.

CAMRS011I INT1SKUT DD OPEN FAILED.

Explanation: The dataset pointed to by the INT1SKUT DD statement cannot be opened. Determine cause and rerun the job.

CAMRAS12I RECORD SKIPPED. INVALID DB TYPE

Explanation: Schedule IMPORT operation found an import record with an invalid TYPE. The record is not imported.

CAMAS13I IMPORT DSN ==> xxxxx

Explanation: xxxxx is the fully qualified dataset containing the Schedule to be imported.

CAMRS14W SCHEDULE IMPORT NON ZERO RC = nn

Explanation: Import operations received a non zero return code of nn. Possible RC from Import:

0 = Import successful

4 = BLSCHTBL either is not anchored or does not exist

8 = INT1SKUT DD statement is missing.

CAMRS15I RECORDS READ FROM INT1SKUT DD: nnnnn

Explanation: During a Schedule Import operation, nnnn records were imported

CAMRS16I BATCH LINK SCHEDULE IMPORTED

Explanation: Informational Message.

CAMRS17I SCHEDULE WILL BE IMPORTED TO REFRESH BLSCHTBL

Explanation: The default refresh option will be used to import the Schedule.

CAMRS18I SCHEDULE WILL BE ADDED TO BLSCHTBL.

Explanation: ADD option specified by the Execution PARM is used to add from external dataset to the current active schedules in CSA.

CAMRS19E INT1SKUT DD OPEN FAILED.

Explanation: Open of dataset pointed to by INT1SKUT DD card failed.

CAMRS20E INT1SKUT DD CARD NOT FOUND.

Explanation: INT1SKUT DD card is missing from the execution JCL. Make sure the DD Name is correctly spelled.

CAMRS20E BATCH LINK SCHEDULE NOT IMPORTED.

Explanation: For reasons stated in previous messages, the Batch Link Schedule was not imported.

CAMRS21I INT1SKUT CLOSED.

Explanation: Informational message.

CAMRS22W BLSCHTBL NOT ANCHORED.

Explanation: The Batch Link Schedule table is not anchored off CAAT. Review the initialization job log to determine cause. Correct error and rerun export operation.

CAMRS23I SP241 ACQUIRED AT ADDR ==> xxxxxxxx

Explanation: Information. Subpool 241 at address xxxxxxxx was acquired for Batch Link Schedule in-core table.

CAMRS24E #CAAT ADD CA ANCHOR FAILED. RC=nn

Explanation: The #CAAT facility to add Batch Link Schedule table failed. Return code is nn. Make sure the initialization program, MR91INIT, is executing out of an APF authorized library.

CAMRS25W SP241 GETMAIN FAILED.

Explanation: GETMAIN for Subpool 241 storage failed. This could be caused by excessive number of entries defined in BLSCHTBZ. Each entry requires 27 bytes. Another likely cause - your CSA is over utilized. Discuss this issue with your Systems Programmer.

CAMRS26I ACQUIRED SP241

Explanation: Informational message.

CAMRS27I CA ANCHORED ADDED.

Explanation: Informational. The Batch Link Schedule table was successfully anchored by the initialization routine.

CAMRS28W CAMBSTBZ LOAD FAILED. SCHEDULING DISABLED.

Explanation: LOAD of CAMBSTBZ table failed, Batch Link Scheduling facility has been disabled. Make sure the load library containing CAMRBSTBZ is in the STEPLIB concatenation stream. Rerun the initialization routine.

CAMRS29I MAXIMUM SCHEDULE ENTRIES DEFINED: nnn

Explanation: Your installation has defined nnn as the maximum number of schedule entries that can be defined in the schedule table.

CAMRS29W 0 ENTRIES DEFINED. SCHEDULING DISABLED.

Explanation: Informational. CAMBSTBZ defined 0 entries for Batch Link Schedule table, resulting in the Schedule facility being disabled. To enable the Schedule facility, change table CAMBSTBZ. See the *Installation Guide* for instructions on modifying this table. Rerun InterTest Batch initialization routine.

CAMRS30I NUMBER OF SCHEDULE ENTRIES IMPORTED: nnn

Explanation: The total number of entries successfully imported is nnn entries.

CAMRS31W INSUFFICIENT SCHEDULE TABLE SIZE. nn SCHEDULE ENTRIES NOT IMPORTED.

Explanation: There were *nn* entries from the INT1SKUT data set not imported due to the restriction defined in the Batch Link Scheduling Table (CAMRSTBZ). A return code of 12 was also raised. To review the entries that were imported, select Option 1 from the CA InterTest Batch DB2 and IMS Schedule Menu.

CAMRS32W BLSCHTBL IS FULL. CANNOT ADD ANY MORE.

Explanation: The In Core Schedule Table is full. Additional schedule entries cannot be added. A return code of 4 indicated some of the entries from the INT1SKUT data set were added. A return code of 8 indicated that none of the entries from the INT1SKUT were added. You can display the contents of the schedule table by selecting Option 1 from the CA InterTest Batch DB2 and IMS Schedule Menu.

CAMRS33E SLOT NUMBER REQUESTED > MAXIMUM SIZE.

Explanation: Internal schedule component error. Contact CA for assistance.

CAMRS34E BLSCHTBL SLOT NUMBER ERROR. FOUND=> *nnnn.* IN SLOT NUMBER => *nnnn.*

Explanation: Internal schedule component error. Contact CA for assistance.

INTA006 DATA SET NOT CATALOGED

Explanation: The data set requested on the panel is not a cataloged data set. Check the name of your fully qualified data set and retry.

INTA007 DATA SET IS AN ALIAS

INTA008 CATALOG DOES NOT EXIST

INTA009 SYNTAX ERROR IN DSNAME

INTA010 INVALID DATA SET NAME

Explanation: The data set name entered has invalid syntax.

INTA020 CA PANVALET LIB INVALID

Explanation: Only OS libraries can be specified for output.

INTA021 ENTER REQUIRED FIELD

Explanation: Enter required field at the cursor.

INTA110 INVALID DATA SET NAME

Explanation: The data set name entered has invalid syntax.

INTA120 NO DD STATS TRANSLATED

Explanation: The step selected contained no DD statements to translate.

INTA130 STEP NOT FOUND

Explanation: The input JCL either does not contain an exec card, or the step requested on the input specification panel cannot be found in the member specified.

INTA210 INVALID DATA SET ORG

Explanation: The data set name entered has invalid syntax.

INTA211 INVALID RECORD FORMAT

Explanation: The record format is not fixed.

INTA212 INVALID RECORD LENGTH

Explanation: The record length must be 80.

INTA220 MISSING DDNAME

Explanation: Ddname must be specified at the cursor position.

INTA221 NO DSN, TERM, OR SYSOUT

Explanation: Enter a data set name, terminal, or SYSOUT at the cursor.

INTA222 INVALID DATA SET NAME

Explanation: Check data set name.

INTA223 INVALID DISPOSITION

Explanation: The DISP is incorrect. Valid DISP entries are SHR, OLD, and NEW.

INTA224 INVALID COMMAND

Explanation: An invalid command was entered on the command line.

INTA225 INVALID RECORD FORMAT

Explanation: Check record format.

INTA226 INVALID DSORG

Explanation: Use a DSORG of PS or PO.

INTA300 MSL BUILD ERROR PDS ERR

INTA301 MSL BUILD ERROR PAN ERR

Explanation: A CA Panvalet read error or access error may have occurred.

INTA302 MEMBER NOT FOUND

Explanation: The member name requested on the panel cannot be found in the specified PDS.

INTA303 NOT A PDS OR PANLIB

Explanation: Input file is neither a PDS nor a PANLIB.

INTA304 MEMBER NOT FOUND/PRESENT

Explanation: Selected member was not found or not present during ALIB I/O.

INTA305 I/O ERROR ON DIRECTORY

Explanation: An I/O error occurred during the read of a PDS directory.

INTA306 INPUT FILE NOT PRE-ALLOC

Explanation: The input file was not pre-allocated or the allocation failed.

INTA307 INPUT FILE OPEN ERROR

Explanation: The input PDS you have entered is an ALIB, with a RECL=150. The input data set can be either a JCL PDS or a CA PANVALET LIB.

INTA308 OUT FILE NOT PRE-ALLOC

Explanation: The output file was not pre-allocated or the allocation failed.

INTA309 OUTPUT FILE OPEN ERROR

Explanation: The output PDS you have entered as an ALIB is not defined correctly. An ALIB PDS must be defined as RECL=150.

INTA310 INVALID ACCESS CODE

Explanation: Enter this member's correct access code.

INTA311 MEMBER NOT READ

Explanation: An internal error occurred when trying to read member.

INTA312 INVALID CONTROL CODE

Explanation: Enter this library's correct control code.

INTA510 ALLOCATION FAILED

Explanation: The browse temporary data set could not be allocated.

INTA520 DCB OPEN FAILED

Explanation: The browse temporary data set could not be opened.

INTA610 CLIST ERROR HAS OCCURED

INTA620 CLIST WRITE ERROR

Explanation: An error occurred during CLIST file write.

INTA621 CLIST BUILD ERROR

Explanation: An error occurred during CLIST build.

INTAK210 DATA SET NOT AVAILABLE

Explanation: The DISP specified is requesting exclusive control and the data set may already be in use.

INTAK218 VOLUME IS NOT MOUNTED

Explanation: The volume specified in the JCL is not active. Modify the volume and retry allocation

INTAK219 UNIT NAME UNDEFINED

Explanation: The specified unit name is undefined in DSNAME allocation.

INTAK350 INVALID PARM IN TEXT

Explanation: There is an unacceptable parameter in the JCL. The JCL being used is not valid for execution in batch. Modify the JCL and retry allocation.

INTAK360 INVALID KEY IN TEXT

Explanation: Invalid key specified in text unit (all SVC 99 functions).

INTAK364 DSNAME ALLOCATION ERROR

Explanation: JOBLIB, STEPLIB, JOBCAT, and STEPCAT are not valid ddnames for TSO.

INTAK368 UNAUTHORIZED USER

Explanation: An authorized function was requested by an unauthorized user.

INTAK369 INVALID PARAMETER LIST

Explanation: Invalid parameter list format (all SVC 99 functions).

INTAK410 DDNAME UNAVAILABLE

Explanation: One of the ddnames being allocated is already defined to your TSO session. Free the DD and try to allocate again.

INTAK420 DDNAME IS OPEN

INTAK438 DDNAME NOT ALLOCATED

INTAK448 DATA SET ALREADY EXISTS

Explanation: Occurs when attempting to allocate a data set with a DISP=NEW and the data set already exists.

INTAL708 DATA SET NOT CATALOGED

Explanation: One of the data sets in your JCL is defined with a DISP=OLD or SHR and the data set is not cataloged.

INTAM100 INVALID OPTION

Explanation: Options 1, 2, 3, 4, and 5 are the only options available.

INTAM704 DUPLICATE DSN ON VOL

INTAM714 SPACE NOT AVAILABLE

Explanation: The JCL you are attempting to allocate contains a new data set, and the volume specified for that data set does not have sufficient space to allocate it. Specify another volume.

INTAN708 CATALOG ERROR

Explanation: Structure inconsistent.

INTB000 FUNCTION FAILED

INTB001 CATALOG I/O ERROR

INTB002 CATALOG I/O ERROR

INTB003 INVALID MEMBER NAME

INTB004 INVALID DATA SET NAME

INTB005 VOLUME NOT AVAILABLE

INTB006 DAIR ERROR

Explanation: Error occurred using Dynamic Allocation Interface Request.

INTB007 DAIR CATALOG ERROR

INTB008 DAIR ALLOCATE ERROR

INTB009 DAIR FREE ERROR

Explanation: Dynamic Allocation Interface Request attempted to free a data set via ddname unsuccessfully.

INTB100 NOTHING TO ALLOCATE

Explanation: Allocate was requested but no ddnames existed to allocate.

INTB101 REQUEST BLOCK FAILURE

Explanation: GETMAIN failed to get storage for request block in SVC99.

INTB102 PAGE FAILED FOR TEXTUNIT

Explanation: GETMAIN failed to get a page for SVC99 text units and PTR

INTG001 INVALID LINE COMMAND

INTG002 NUMBER IS INVALID

Explanation: A number is not allowed with the “AL” line command

INTG003 MEMBER NOT FOUND

Explanation: Selected member was not found.

INTG005 INVALID PRIMARY COMMAND

Explanation: Enter UP, DOWN, END, RETURN, SELECT, or LOCATE.

INTG007 NO OPERATOR FOUND

Explanation: Operator for command not present or valid.

INTG008 MEMBER IS SAVED

Explanation: Intaenter has overwritten the input member and DSN

INTG009 MEMBER IS CREATED

Explanation: Intaenter has written to a new member or DSN.

INTG101 COMMAND NOT FOUND

Explanation: Command is either not found or miskeyed.

INTG102 MEM REPLACE NOT ALLOWED

Explanation: REPLACE NO was specified on out panel. Use Y.

XDAIR004 INVALID PARM LIST

Explanation: The parameter list passed to DAIR was invalid.

XDAIR008 CATALOG ERROR

Explanation: An error occurred in the catalog management routine.

XDAIR016 NO TIOT ENTRIES

Explanation: No TIOT entries are available for use.

XDAIR020 DSNAME UNAVAILABLE

Explanation: The data set requested is not available.

XDAIR024 DSNAME IS CONCATENATED

Explanation: DSNAME requested is a member of a concatenated group.

XDAIR028 DSNAME IS NOT ALLOCATED

Explanation: DSN is not allocated or its attribute list is not found.

XDAIR032 DSN PREVIOUSLY ALLOCATED

Explanation: DSN previously allocated permanently, DISP=NEW not allowed.

XDAIR036 ERROR OCCURRED IN CATLG

Explanation: An error occurred on a catalog information routine (IKJEHCIR)

XDAIR040 RETURN AREA EXHAUSTED

Explanation: The provided return area is exhausted and more index blocks exist.

XDAIR044 DELETE PREVIOUSLY USED

Explanation: You specified DELETE or a request of OLD, MOD, or SHR with no volume.

XDAIR048 RESERVED

XDAIR052 REQUEST DENIED

Explanation: Request has been denied by installation exit.

System Abends

This subject contains some system ABEND codes and their probable causes that may occur while debugging your program. The information is given here for your convenience. However, the definitive sources for these codes are the following IBM manuals: *System Codes* (GC38-1002) and VS2 *System Messages* (GC38-1003) or *MVS/XA System Codes* (GC28-1157) and *MVS/XA System Messages*, Volumes 1 (GC28-1376) and 2 (GC38-1377).

001 I/O error.

Attempt to read after AT END.

Reading a wrong length record.

002 I/O error.

003, 004, and 008 I/O error.

013	Something does not match up between the program and the DD statement. Check the system message (IEC141I) for a more complete description of the error.
02x	Conflicting DCB for a direct file
030	Error in DCB for ISAM file.
031	I/O error on an ISAM file.
032	Error in DCB for ISAM file.
033	I/O error on an ISAM file.
034	Error in DCB for ISAM file.
035	Error in DCB for ISAM file.
036	No prime data area was specified for an ISAM file.
037	Invalid buffer length was specified.
038	The index area is too small.
039	END-OF-DATA problem with an ISAM file.
03A	I/O error on an ISAM file.
03B	Opening an ISAM file that is already open.
03C	Error in DCB for an ISAM file.
03D	Conflict in DCB for ISAM file - the DCB=DSORG=IS was omitted.
03E	I/O error during the load of an ISAM file.
0C1, 0C2, and 0C3	Invalid or missing DD statement. Reading or writing an unopened file. Runaway or uninitialized subscript or index. Dropping out of end of program.
0C4 and 0C6	Invalid or missing DD statement. Reading or writing an unopened file. Runaway or uninitialized subscript or index. Dropping out of end of program. Opening or closing a file twice. Coding equal BLKSIZE and LRECL for variable length records. Omitting a parameter from a CALL list. Using a divisor or multiplier that is too large. Using the wrong SORTLIB in a COBOL sort.

0C7	Data item not initialized. DISPLAY data moved to a numeric group item. Improper definition in LINKAGE SECTION. Improper definition in FILE SECTION. Invalid or improperly aligned data in an input file. Improper initialization using MOVE ZERO. Runaway or uninitialized subscript or index.
0C8 and 0CA	Arithmetic overflow: the receiving field of an arithmetic operation is too small.
0C9 and 0CB	Divide exception: dividing by zero.
0CC, 0CD, 0CE, and OCF	A problem in floating point operation. Did you specify COMP-1 or COMP-2 usage accidentally?
13E in BMP Region	CA InterTest Batch END, QUIT, or CANCEL caused a detach error in the BMP region.
213	The system could not find the file as specified in the allocate statement. Check system message for further explanation.
437	An I/O error occurred while processing the end-of-volume condition.
706	Attempt to execute a program that has been marked 'not executable' by the linkage editor. Attempt to execute an object file. Attempt to execute a source file.
806	The load module was not found in the system library or the dynamic STEPLIB you provided.
80A	Not enough virtual storage was available to allocate the amount required by the program. Log on specifying a larger region size and retry. Some data centers have TSO size restrictions. If you specify a size larger than permitted, the size allocated will default to the data center's default size.
90A	An area of main storage was released by a program, but it was an area that was not attached to the program. This usually means a file was closed twice.
913	A security or password violation has occurred.
B37	The file required more direct access space than was requested by the DD statement. This is usually caused by incorrect direct access space allocation or a program loop that contains a WRITE statement.
C03	The application that you were debugging has left some files open. Correct the logic in your application to close the files before exiting.

D37	The file required more direct access space than was requested by the DD statement. This is usually caused by an underestimated direct access space allocation or a program loop that contains a WRITE statement.
E37	The volume does not contain enough available space to allocate the amount requested for a file.

User Codes

519	A procedure exit has been reached, and the user has bypassed statements so the references to the calling statement are not defined.

Glossary

ALIB (Allocation Library)

An Allocation Library is the internal file allocations format of the application. The ALIB is a partitioned data set that consists of members created through the ALIB Editor or JCL Converter.

CLIST (Command List)

A Command List is an executable sequence of TSO commands, subcommands, and command procedure statements.

Conditional Breakpoint

A Conditional Breakpoint stops program execution at a specified statement when the WHEN command statement is true. The W when-name Intercept panel is then displayed. These breakpoints are controlled with the WHEN and OFFWHN commands.

Frequency Counter

The Frequency Counter counts the number of times a statement has executed. Frequency counting can be turned on or off with the FREQ command.

Global Conditional Breakpoint

The Global Conditional Breakpoint instructs the application to test for a specified condition before every executable statement. The application halts the program before executing the statement that would cause the condition to test true. Global conditional breakpoint are set on the When panel. Compare with Local Conditional Breakpoint

Intercept or Breakpoint

An intercept/breakpoint stops the test and displays the executable statement at the point of intercept. The next statement to be executed is highlighted. This panel is called the Intercept panel and is displayed by any of these conditions:

- Initial entry
- Program abend
- Pressing the Attention key
- A breakpoint has been encountered: (a) Conditional, (b) Unconditional, (c) at entry, (d) at exit
- STEP command
- NEXT command

Line Command

A Line Command is a command entered in the left-most field on a line of the program listing while at an Intercept panel.

Load Module

A Load Module is an entity processed by the linkage editor to produce a program that is executable by the operating system.

Local Conditional Breakpoint

A Local Conditional Breakpoint instructs the application to test for a specified condition only before execution of a particular statement. The application halts the program before executing the specified statement if execution would cause the local condition to test true. Local conditional breakpoints are set with the C or W line commands. Compare with Global Conditional Breakpoint.

Proclib

Proclib is the Procedure Library, a library used for expanding procedures found in JCL during conversion.

Program-ID

The Program-ID is used by the COBOL compiler to identify the program and link edit it to the load module. It is specified in the IDENTIFICATION DIVISION of the COBOL program.

PROTSYM

The PROTSYM is the VSAM file that contains the symbolic information and source listings for your programs.

Step Count

The Step Count is used with the STEP command. It specifies the number of verbs to be executed when the GO command is entered. When the count is reached, the Step Count Intercept panel is displayed.

Unconditional Breakpoint

An Unconditional Breakpoint stops program execution at a specified statement whenever the statement is encountered and displays the Breakpoint Intercept panel. These breakpoints are controlled with the AT and OFF commands and the A (or U) and O (or X) commands.

Variable Change Breakpoint

A Variable Change Breakpoint instructs the application to halt program execution if the value of a specified data item changes. Variable change breakpoints can be set with the V line command or on the When panel.

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X