

CA InterTest™ and CA SymDump® - 11.0

Getting Started

Date: 06-Jun-2018



This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the “Documentation”) is for your informational purposes only and is subject to change or withdrawal by CA at any time. This Documentation is proprietary information of CA and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA.

If you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2018 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Table of Contents

CA InterTest for CICS Primers	11
Assembler Primer	11
Getting Help	11
Assembler Basic Demo Session	11
Demo Session Objectives	12
Select the Source Program from the ITST Menus	12
View the Source Listing Profile	15
Set Monitoring	16
View Monitoring Status	17
Start Program Execution	18
Detect and Prevent an Abend	19
Determine the Cause of the Error	20
Change the Value in TASKNUM Dynamically	21
Structured Storage Display	21
Control Program Execution	23
Set Unconditional Breakpoints	24
Remove the Keep Window and Breakpoints	25
Remove Breakpoints from the Status Display	26
Resume Program Execution	27
Assembler Debugging Programs	29
Checklist of Basic Debugging Tasks	29
Display Your Source Listing	30
Start a Test Session	31
Set and Remove Breakpoints	32
Inspect and Modify Main Storage	34
Keep Variables in the Keep Window	36
Inspect and Modify Auxiliary Storage	40
Resume Program Execution	43
Abend a Task	45
Get Help	45
End a Test Session	46
Assembler Advanced Monitoring Features	47
Set Options from the Monitoring Menus	47
Conditional Breakpoints	49
Request Breakpoints	51
Backtrace Facility	52
Set the Code Counting Coverage Option	55

Replacement, Protection, and Special Options	56
Composite Module Testing	58
Dump Analysis with CA SymDump for CICS	59
Assembler Advanced Demo Session	60
Startup	61
Perform the following steps before initiating the advanced demo session:	61
Execute the Demo Program	67
Option 01 Replace a File Control ID	68
Option 02 Limit CICS Storage and Requests	72
Option 03 Prevent a Program from Updating a File	77
COBOL Primer	84
Getting Help	84
COBOL Basic Demo Session	84
Maintain Synchronized Processing	85
Demo Session Objectives	86
Demo Session Scenario	86
Select the Source Program from the ITST Menus	88
View the Source Listing Profile	91
Set Monitoring	91
View Monitoring Status	92
Start Program Execution	93
Detect and Prevent an Abend	93
Determine the Cause of the Error	94
Dynamically Change the Value in TASKNUM	95
COBOL Control Program Execution	97
COBOL Set Unconditional Breakpoints	98
What You Have Learned	100
COBOL Debugging Programs	101
Checklist of Basic Debugging Tasks	102
Display Your Source Listing	102
Start a Test Session	103
Set and Remove Breakpoints	105
Inspect and Modify Main Storage	109
Keeping Data Items in the Keep Window	112
Use Variable-Change Breakpoints to Detect Changing Values	113
Inspect and Modify Auxiliary Storage	116
Resume Program Execution	119
Abend a Task	120
Get Help	121
End a Test Session	122
Correct the Source Code	122
COBOL Advanced Monitoring Features	123

Set Options from the Monitoring Menus	123
Set and Remove Conditional Breakpoints	124
Remove Conditional Breakpoints from the Source Listing	125
Request Breakpoints	126
Backtrace Facility	128
Set the Code Counting Coverage Option	131
Statement Trace Facility	132
Indirect Commands	134
Replacement, Protection, and Special Options	138
Composite Module Testing	139
Dump Analysis with CA SymDump	140
COBOL Advanced Demo Session	141
Demo Preliminaries	142
Execute the Demo Program	148
Option 01 Replace a File Control ID	149
Option 02 Limit CICS Storage and Requests	153
Option 03 Prevent a Program from Updating a File	157
Option 04 How to Display Variable Length Data	163
Option 05 How to Work with Indexed Table Items	168
Option 06 How to Detect a Storage Violation	173
Option 07 How to Test a Composite Module	176
PL/I Primer	184
Getting Help	184
PL/I Basic Demo Session	184
Maintain Synchronized Processing	185
Demo Session Objectives	186
Demo Session Scenario	186
Select the Source Program from the ITST Menus	187
View the Source Listing Profile	190
Set Monitoring	190
View Monitoring Status	191
Begin Program Execution	192
Detect and Prevent an Abend	193
Determine the Cause of the Error	193
Change the Value in TASKNUM Dynamically	194
Control Program Execution	196
Set Unconditional Breakpoints	197
What You Learned	200
PL/I Debugging Programs	201
Basic Debugging Tasks Checklist	201
Display Your Source Listing	202
Start a Test Session	203

Set and Remove Breakpoints	204
Inspect and Modify Main Storage	208
Keep Variables in the Keep Window	210
Work with Arrays and Qualified Variables	212
Inspect and Modify Auxiliary Storage	213
Resume Program Execution	216
Abend a Task	218
Get Help	218
End a Test Session	220
Correct the Source Code	220
PL/I Advanced Monitoring Features	220
Set Options from the Monitoring Menus	221
Set and Remove Conditional Breakpoints	222
Request Breakpoints	224
The Backtrace Facility	226
Set the Code Counting Coverage Option	229
Replacement, Protection, and Special Options	235
Composite Module Testing	236
Dump Analysis with CA SymDump	237
PL/I Advanced Demo Session	239
Demo Preliminaries	240
Execute the Demo Program	246
Option 01 Replace a File Control ID	247
Option 02 Limit CICS Storage and Requests	251
Option 03 Prevent a Program from Updating a File	255
Option 04 How to Detect a Storage Violation	261
Option 05 How to Test a Composite Module	264

CA InterTest Batch Demo Sessions 273

Getting Help	273
Basic Foreground Demo Session	273
Demo Session Objectives	273
Before You Start the Demo	274
Access the Product	274
The Demo Source	274
CA Roscoe Split Screen Sequence	274
Begin the Basic Demo	274
Access the Primary Option Menu	275
File Allocation	276
Specify the Program to be Tested	276
Turn the Frequency Display On	278

Begin Program Execution	278
Detect and Intercept an Abend	278
Determine the Cause of the Error	279
Dynamically Change the Value in Tasknum	280
Control Program Execution	281
Stop Your Program by Setting Breakpoints	281
Set Unconditional Breakpoints	282
Remove the Breakpoint and Keep Window	284
Continue Execution	285
What You Have Learned	285
Basic Batch Link Demo	285
Demo Session Objectives	286
Before You Begin	286
Begin the Batch Link Demo	286
Test Your Programs Using Batch Link	288
What You Have Learned	289
Advanced Demo Session	289
Advanced Demo Preliminaries	290
Access the Initial Intercept Panel for the Demo Program	290
Split Your Screen	291
Advanced Demo Data Sets	291
INT1CLIB	293
Advanced Demo Session for COBOL	294
Include Unconditional Breakpoints	294
Access the Demo Session Options Menu	295
Exit the Advanced Options Demo	295
Return to the Advanced Options Demo Session	296
Option 1 Time-Controlled Execution	296
Option 2 Set Conditional Breakpoints	298
Option 3 Update Source Code	300
Option 5 Trace Program Execution	307
Option 6 Work with Indexed Table Items	310
Option 7 Histogram Report	313
Advanced Demo Session for Assembler	317
Include Unconditional Breakpoints	317
Access the Demo Session Options Menu	318
Exit the Advanced Options Demo	318
Return to the Advanced Options Demo Session	319
Option 1 Time-Controlled Execution	319
Option 2 Set Conditional Breakpoints	321
Option 3 Update Source Code	323
Option 4 Interrupt a Looping Program	324

Option 5 Trace Program Execution	328
Option 7 Histogram Report	329
Advanced Demo Session for PL/I	331
Include Unconditional Breakpoints	331
Access the Demo Session Options Menu	332
Exit the Advanced Options Demo	333
Option 1 Time-Controlled Execution	333
Option 2 Set Conditional Breakpoints	335
Option 3 Update Source Code	338
Option 4 Interrupt a Looping Program	339
Option 5 Trace Program Execution	345
Option 5 Trace Program Execution	348
Option 6 Work with Indexed Table Items	350
Option 7 Histogram Report	354

Quick References	358
------------------------	-----

Getting Started

This section contains information that will help you get started using the CA InterTest products. The Assembler, COBOL, and PL/I primers walk you through the basic and advanced features of CA InterTest for CICS. You can use the basic and advanced demo sessions for CA InterTest Batch as described here to learn more about how to use the batch tools.

This section also contains quick reference cards for the CA InterTest products that you can view, download, or print.

CA InterTest for CICS Primers

The best way to learn about CA InterTest for CICS is by using the COBOL, PL/I, and Assembler primers. This section teaches you how to use the basic and advanced product features available to you in CA InterTest for CICS. Use the demo sessions for step-by-step instructions on how to use these features. For complete information on all of the product's features and functionality, see [Using CICS Tools \(https://docops.ca.com/display/CAITSD11/Using+CICS+Tools\)](https://docops.ca.com/display/CAITSD11/Using+CICS+Tools).

Assembler Primer

This section provides information designed to train new users in the basic product features used to test and debug programs. This article also introduces some of the advanced features of CA InterTest for CICS.

We recommend that all users take the demo sessions in this article and become familiar with the features explained here. For complete information about these and other features, see [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging). We suggest that you read the [Assembler Basic Demo Session \(see page 11\)](#) first, because it is the best way to begin learning about your product.

Getting Help

The Help facility is online documentation of product features. It makes it easy to learn and to use the product. Help is available from all product screens.

To view an online summary of new features for this release, use ITST Option 8 What's New.

Assembler Basic Demo Session

This article 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 application.

The basic demo session shows many of the testing and debugging tasks you will use on your own programs. For more information on any of these topics, see [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging).

We are assuming that the product is installed on your system. If not, contact your system programmer.

- [Demo Session Objectives \(see page 12\)](#)
- [Select the Source Program from the ITST Menus \(see page 12\)](#)
- [View the Source Listing Profile \(see page 15\)](#)
- [Set Monitoring \(see page 16\)](#)
- [View Monitoring Status \(see page 17\)](#)

- Start Program Execution (see page 18)
- Detect and Prevent an Abend (see page 19)
- Determine the Cause of the Error (see page 20)
- Change the Value in TASKNUM Dynamically (see page 21)
- Structured Storage Display (see page 21)
- Control Program Execution (see page 23)
- Set Unconditional Breakpoints (see page 24)
- Remove the Keep Window and Breakpoints (see page 25)
- Remove Breakpoints from the Status Display (see page 26)
- Resume Program Execution (see page 27)

Demo Session Objectives

The demo session teaches you how to:

- Select a source listing for a program from the ITST menus
- Inform the application that you want to test a program (set monitoring)
- Respond to the information provided when a program error is detected
- Display the value of a program variable
- Dynamically change the value of a program variable
- Halt program execution at any point (set breakpoints)
- Resume program execution

Select the Source Program from the ITST Menus

Begin your session by selecting the source listing of the ASMDemo demonstration program from the ITST Primary Option Menu.

1. Sign on to CICS. Type **ITST** on a clear screen.
2. Press Enter.
The Primary Option Menu appears.

```

CA InterTest for CICS PRIMARY OPTION MENU
OPTION ===>

 1 Source           - Display/select program source files/listings
 2 Monitoring       - Display/modify CA InterTest monitoring/activity
 3 Main storage     - Display/modify CICS storage areas
 4 Auxiliary storage - Display/access databases/files/queues
 5 Dump analysis    - Invoke CA SymDump CICS dump/trace capture facility
 6 Product help     - Invoke CA InterTest product help facility
 7 Status/Maintenance - Product status and maintenance functions
 8 What's new?      - Display information about CA InterTest for CICS
 X Exit            - Terminate menu processing

```

PF1 Help	2	3 End	4 Return	5	6
PF7	8	9	10	11	12

Notice the PF keys have conventional assignments for ISPF-like navigation:

- PF1 Help accesses online help
- PF3 End displays the previous panel
- PF4 Return returns to the top-level menu

Now we can take a look at the source of the program we are debugging.

1. Type **1** in the Option field.

2. Press Enter.
The Source Menu displays.

```

----- CA InterTest for CICS SOURCE MENU -----
OPTION ==> 1

Select a member list type, specifying optional criteria below.

1 Source listings - Display/select program source listings
2 Symbolic files - Display/select program source SYMBOLIC files

Type specific or generic program/file name(s):
(Valid mask characters are * and/or +)

asm* -----
    
```

3. Type **1** in the Option field to search for program listings.

4. Tab to the first entry field for program/file names, and type the full or partial name of the program to be monitored. Try using a mask entry such as **a***, **as***, or **asm***. The asterisk is a generic or wildcard character indicating that anything from this point on in a file name satisfies the search criteria.

5. Press Enter.
The Source Listing Selection screen appears. It lists all program Source Listings meeting your search criteria on the previous menu. In our example, the list shows all programs beginning with the letters **asm**, in all Symbolic files. Your list might have fewer or more entries.

```

----- CA InterTest for CICS SOURCE LISTING SELECTION -----
COMMAND ==>

Type S to select a source listing.

Name      File      Created      Size Attributes
- ASMDemo PROTSYM 08/16/2013 15:29 74 HLASM 3.0, no purge
- *** End of data ***

PF1 Help      2 Refresh    3 End        4 Return     5            6
PF7 Backward  8 Forward    9            10           11           12
    
```

6. Locate the demo program, ASMDemo, in the Selection List. You might need to scroll the file list using PF7 and PF8.



Note: If the file list does not contain the ASMDemo demo program indicated previously, it could be because the person who did the installation changed the name of the sample programs. Check with that person to find out the correct names.

- Type **S** in the field to the left of the program ASMDemo. Press Enter.
The PROTSYM File Source Listing Display appears, it shows the compiled listing of ASMDemo.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>
Program= ASMDemo Option #      Stmt #      Displacement=      Margin= 01
                        Search=
-----+-----
      Active Usings: None

LOC  OBJECT CODE  ADDR1 ADDR2  STMT      SOURCE STATEMENT
    00000          1 R0      EQU      0
    00001          2 R1      EQU      1
    00002          3 R2      EQU      2
    00003          4 R3      EQU      3

    00004          5 R4      EQU      4

    00005          6 R5      EQU      5
    00006          7 R6      EQU      6

    00007          8 R7      EQU      7

    00008          9 R8      EQU      8
    00009         10 R9      EQU      9
    0000A         11 R10     EQU     10

    0000B         12 R11     EQU     11

    0000C         13 R12     EQU     12

    0000D         14 R13     EQU     13

    0000E         15 R14     EQU     14

    0000F         16 R15     EQU     15
    
```

- The top section of the display identifies the program and provides fields for entering commands, options, statement numbers, and search criteria. Depending on your defaults, it may also display PF key assignments and available options. These have been omitted from the Source Listing displays in this section for clarity.
- The bottom section contains the source listing for ASMDemo.

 **Note:** The statement numbers on your screen might not match those shown in the examples. This does not hinder you from performing the test session.

View the Source Listing Profile

Depending on the configuration settings, the PF key assignments for the Source Listing display might not be shown at the top of the screen. If they are not, type **PROFILE** on the Command line and press Enter (or, press PF4 Profile) to view the various options and PF key functions available from the Source Listing display.

The Source Listing Profile for your current session appears. This lists the PF keys, options, and current environment settings relating to the Source Listing display.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING PROFILE
COMMAND ===>
Program= ASMDEMO Option # Stmt # Displacement= Margin= 01
Search=
OPTS 1 1st CSECT 2 3 4 5 Macro Cat 6 Xref
7 Literals 8 Err msgs 9 Srch fwd 10 Srch bwd 11 12 Bkpt opts
PFKS 1 Help 2 3 End 4 Auto prms 5 Monitor 6 Menu
7 Backward 8 Forward 9 Next Wnd 10 11 12 Status
-----+-----
Display window = N N (None), T (Titles), R (Registers),
K (Keep), P (Program)
PF7/8 amount = PAGE PAGE, HALF, STOP, or a number from 1 to 9999
Stepping amount = 001 The number of INSTR to execute
Auto-stepping = OFF ON to activate; press PF4 to change values
Source List BKPT = ON OFF to use the detailed breakpoint display
From terminal ID = G001 Terminal ID where the program will execute
BKPT terminal ID = G001 Terminal ID to receive the breakpoint displays
User ID = .ANY User ID who will execute this program
AutoKeep Display = OFF ON to activate
Code Counting = OFF ON to activate Code Coverage
    
```



Note: If you want the four lines labeled OPTS and PFKS to appear at the top of your screen throughout your test session, change the Display window field from N to T. Enter **R** in the Display window field if you want to have the contents of the registers shown at the top of the breakpoint display, as shown in the following screen:

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING BREAKPOINT
COMMAND ===>
Program= ASMDEMO Option # Stmt # Displacement= Margin= 01
Search=
-----+-----
R0-R7 0DB053E6 00180C10 0DB052F8 0DB062F8 0DB072F8 000001D5 00000232 0DB082F8
R8-R15 00000000 00000000 00055380 001800D0 008BD000 00180BA8 8DB053C4 00000000
Cond. Code = 0 Amode = 31
-----+-----
    
```

Set Monitoring

The next step is to instruct the application to monitor ASMDEMO. As part of monitoring a CICS command-level program such as ASMDEMO, the application automatically prevents all CICS abends during execution. How this works is shown later when the application prevents ASMDEMO from abending because of an ASRA.

There are several ways you can tell the application to monitor the program. If you are currently viewing a program's source listing, you can:

- Type **MONITOR** on the Command line and press Enter.
- Press PF5 (notice PF5 is listed as Monitor in the earlier illustration).

- Press PF5 or use the MONITOR command while in the Source Listing Display or the Source Listing Profile screen.
The screen momentarily flashes as the request is processed. The Source Listing Display screen then redispays.

View Monitoring Status

To verify that the demo program is being monitored:

1. Type **STATUS** on the command line.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==> status
Program= ASMDEMO Option # Stmt # Displacement= Margin= 01
Search=
-----+-----
Active Usings: None

LOC OBJECT CODE ADDR1 ADDR2 STMT SOURCE STATEMENT
00000 1 R0 EQU 0
00001 2 R1 EQU 1
    
```

2. Press Enter.



Note: PF12 also displays the Monitoring Status display.

The Monitoring Status screen appears.

```

----- CA InterTest for CICS MONITORING STATUS -----
COMMAND ==>

Type + to expand or - collapse option levels displayed below,
or R to remove option(s).

Option Description Attributes
- ASMDEMO Program monitor entry Assembler
- |-.ANY User monitoring options Active
  | Symbolic listing file PROTSYM
- |-SLB Source listing breakpoints G001
  *** End of data ***

PF1 Help 2 Refresh 3 End 4 Return 5 Collapse 6 Expand
PF7 Backward 8 Forward 9 10 11 12
    
```

The monitoring status display gives you an up-to-date summary of how the application is monitoring your program, in an expandable/collapsible tree format. You can scroll the list, collapse or expand it using the indicated PF keys on the bottom of the display. Here is what each entry for ASMDEMO indicates:

- The top entry tells you the application is monitoring the Assembler program ASMDEMO.
- The second entry (.ANY) indicates the application monitors ASMDEMO when any user executes it. If your status display shows your CICS User ID instead of .ANY, that is fine; it indicates monitoring and options taking effect whenever your ID executes the program,

Determine the Cause of the Error

We confirm that the value stored in TASKNUM is not in a valid arithmetic format by displaying its current value. That is, its value prior to the execution of the AP instruction that triggered the ASRA.

1. Overtyping the A with K (keep).
2. Place the cursor under any character in TASKNUM (see the following screen).
3. Press Enter.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING BREAKPOINT
COMMAND ==>
Program= ASMDEMO Option # Stmt # Displacement= Margin= 01
Search=
-----+-----
LOC OBJECT CODE ADDR1 ADDR2 STMT SOURCE STATEMENT
00016A 4050 D1B0 001B0 1355 STH R5,MSGLEN
K 00016E FA20 D189 4AC0 00189 02AC0 ==> AP TASKNUM,=P'1'
==>
==> ASRA ABEND (0C7) detected and prevented. Caused by invalid decimal
==> arithmetic data format.
==>
==> Press PF1 for a detailed description.
==>
- 000174 F920 D189 4AC1 00189 02AC1 1357 CP TASKNUM,=P'2'
- 00017A 4720 2300 00300 1358 BH ENDMSG
- 00017E F920 D189 4AC0 00189 02AC0 1359 CP TASKNUM,=P'1'
- 000184 4780 2256 00256 1360 BE SENDSCR4
- 000188 47F0 2300 00300 1361 B ENDMSG
- 00018C 1363 RETURN DS 0H
1364 * EXEC CICS SEND FROM(OUTMSG
1365 DFHECALL =X'04043000080000
), (FB_2,OUTLEN)
00018C 4110 D068 00068 1381 * EXEC CICS SEND FROM(SCREE

```

The application displays the value of TASKNUM in a specially formatted Keep window, as shown in the following screen.

- On the left is the name of the variable, preceded by the base register
- On the right is its hexadecimal value and corresponding character representation

The application determines the base register from the generated object code of the AP instruction. In this case, it is register 13 (D).

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= ASMDEMO Option # Stmt # Displacement= Margin= 01
Search=
-----+-----
R13.TASKNUM | ?00000.
-----+-----
Loc Object Code Addr1 Addr2 Stmt Source Statement
- 000192 4050 D1B0 001B0 1456 STH R5,MSGLEN
A 000196 FA20 D189 4AE8 00189 02AE8 ==> AP TASKNUM,=P'1'
- 00019C F920 D189 4AE9 00189 02AE9 1458 CP TASKNUM,=P'2'
- 0001A2 4720 2328 00328 1459 BH ENDMSG
- 0001A6 F920 D189 4AE8 00189 02AE8 1460 CP TASKNUM,=P'1'
- 0001AC 4780 227E 0027E 1461 BE SENDSCR4
- 0001B0 47F0 2328 00328 1462 B ENDMSG
- 0001B4 1464 RETURN DS 0H
1465 * EXEC CICS SEND FROM(OUTMSG
1466 DFHECALL =X'04043000080000
), (FB_2,OUTLEN)
0001B4 4110 D068 00068

```

```

                                1482 *      EXEC CICS SEND FROM(SCREEN
                                1483      DFHECALL =X'04043000080000
0001DA 4110 D068              00068      2), (FB 2,MSGLEN)
                                1499 *      EXEC CICS RETURN
                                1500 *      TRANSID(EIBTRNID)
    
```

Any number of variables can be saved in a Keep window. However, only six variables display at a time. PF keys 19 and 20 are provided to scroll forward and backward through the Keep window when it contains more than six variables. This feature lets you see how the variable values change as your program executes. The Keep window remains until you remove all variables from it.

To display the value of a variable without keeping it in a Keep window, type D (display) rather than K (keep). This displays the value of the variable and all variables below it in the same Assembler DSECT.

Now take a look at the contents of TASKNUM. It does not contain a valid packed decimal value, we can see this because there is a "?" in the left most position of the field. Assembler does not allow you to add a value to a field without initializing it.

Change the Value in TASKNUM Dynamically

Now that we have identified and confirmed the cause of the problem, we will fix it.

1. Tab the cursor to the place where the question mark appears.
2. Type **0** over the question mark and erase to end of field (EOF).
3. Press Enter.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= ASMDEMO Option # Stmt # Displacement= Margin= 01
Search=
-----
----- R13.TASKNUM | 0
-----++-----
Loc Object Code Addr1 Addr2 Stmt Source Statement
_ 000192 4050 D1B0 001B0 1456 STH R5,MSGLEN
A 000196 FA20 D189 4AE8 00189 02AE8 ==> AP TASKNUM,=P'1'
_ 00019C F920 D189 4AE9 00189 02AE9 1458 CP TASKNUM,=P'2'
_ 0001A2 4720 2328 00328 1459 BH ENDMSG
_ 0001A6 F920 D189 4AE8 00189 02AE8 1460 CP TASKNUM,=P'1'
_ 0001AC 4780 227E 0027E 1461 BE SENDSCR4
_ 0001B0 47F0 2328 00328 1462 B ENDMSG
_ 0001B4 1464 RETURN DS 0H
1465 * EXEC CICS SEND FROM(OUTMSG
1466 DFHECALL =X'04043000080000
0001B4 4110 D068 00068 ) , (FB 2,OUTLEN)
1482 * EXEC CICS SEND FROM(SCREEN
1483 DFHECALL =X'04043000080000
0001DA 4110 D068 00068 2), (FB 2,MSGLEN)
1499 * EXEC CICS RETURN
1500 * TRANSID(EIBTRNID)
    
```

The application redisplay the screen. TASKNUM now contains a packed decimal zero.

Structured Storage Display

Now we request a structured storage display of TASKNUM and all the fields below it in the same DSECT.

1. Type **D** to the left of TASKNUM in the Keep window, and press Enter.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= ASMDEMO Option # Stmt # Displacement= Margin= 01
Search=
-----
d____ R13.TASKNUM | 00000.
-----
++
Loc Object Code Addr1 Addr2 Stmt Source Statement
_ 000192 4050 D1B0 001B0 1456 STH R5,MSGLEN
A 000196 FA20 D189 4AE8 00189 02AE8 ==> AP TASKNUM,=P'1'
_ 00019C F920 D189 4AE9 00189 02AE9 1458 CP TASKNUM,=P'2'
_ 0001A2 4720 2328 00328 1459 BH ENDMSG
_ 0001A6 F920 D189 4AE8 00189 02AE8 1460 CP TASKNUM,=P'1'
_ 0001AC 4780 227E 0027E 1461 BE SENDSCR4
_ 0001B0 47F0 2328 00328 1462 B ENDMSG
_ 0001B4 1464 RETURN DS 0H
1465 * EXEC CICS SEND FROM(OUTMSG
1466 DFHECALL =X'04043000080000
), (FB 2,OUTLEN)
0001B4 4110 D068 00068 1482 * EXEC CICS SEND FROM(SCREEN
1483 DFHECALL =X'04043000080000
2), (FB 2,MSGLEN)
0001DA 4110 D068 00068 1499 * EXEC CICS RETURN
1500 * TRANSID(EIBTRNID)

```

The following screen appears. On the left are the Assembler labels; on the right are the hexadecimal values and corresponding character representations of each field.

```

CA InterTest for CICS - MAIN STORAGE UTILITY - Termid = U026

Starting at Address = 105018 Disp Structure Display Format
TASKNUM 189 | 00000. |
TSTEXT 18C | ..... |
| ..... |
MSGNAME 1AC | 0000000000. |
MSGLEN 1B0 | +00469. |
RECRBA 1B2 | ..... |
RECPONT 1B8 | 0000000000. |
| 1BC | 00000. |
DMP04AS 1BC | ..... |
| 1BC | ..... |
RECOUT1L 1C8 | .. |
RECOUT1F 1CA | . |
RECOUT1A 1CA | . |
RECOUT1I 1CB | . |
RECOUT1O 1CB | ..... |
-----
PF1 Help 2 3 End 4 Return 5 6 Dump
PF7 Backward 8 Forward 9 Caps Off 10 11 Redisplay 12 Structure
CORE='R13.TASKNUM'
CAIN4713 R13 not found - have assumed its register value

```

The structured format of the main storage display is much easier to use than a traditional dump-formatted display. Since the application displays only those bytes belonging to a field, you do not have to know or remember the lengths of the fields to determine their contents.



Note: You can change the contents of any field by overtyping the desired bytes on the main storage display. By default the application displays appropriate translated values for each field. If you prefer you may toggle to hex/char display mode by hitting the PF12 key, data alteration is allowed in either mode.

Now we return to the program listing so we can continue to test ASMDemo.

2. Press Clear or PF3.
The breakpoint screen redisplay.

Control Program Execution

Now that the value in TASKNUM has been properly initialized, the next step might be to continue testing by resuming program execution. However, this is a good opportunity to learn about an important feature -- the ability to control program execution by setting breakpoints.

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 task, and the program either runs to completion or abends.

With the application you can control program execution in a number of ways. For example, you can set stops, called *breakpoints*, anywhere in your program. Four types of breakpoints you can set are unconditional, conditional, variable-change, and request breakpoints.

- When you set an *unconditional* breakpoint at an instruction, the program stops just before or just after the instruction is executed. (Depending on the type of unconditional breakpoint that you set.)
- When you set a *conditional* breakpoint at an instruction, the program stops only if a condition you specified is met, such as a counter equaling or exceeding some value. You can also set a conditional breakpoint to stop at any instruction when the condition you specified is met.
- When you set a *variable-change* breakpoint, the program stops at any instruction if the value of the variable you specify has changed. This is a special type of conditional breakpoint.
- When you set a *request* breakpoint, the program stops before all CICS commands and macros and other program calls (such as calls to DL/I or DB2) or just before specific CICS commands (such as all READ or WRITE commands)

What You Can Do When Your Program Is Stopped

Once a program is stopped, you can use the testing and debugging facilities to:

- Examine the source listing
- Examine and modify main and auxiliary storage to detect and correct errors
- Set and remove breakpoints
- Keep variables in a Keep window to observe changes in their values
- Abend your task with or without a dump
- Go around a problem by resuming program 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 just one instruction and then stops

Now we are going to demonstrate how easy it is to control program execution by setting an unconditional breakpoint.

Set Unconditional Breakpoints

You can set unconditional breakpoints directly on the Source Listing screen just as easily as you displayed and modified TASKNUM . Enter U or) to the left of each instruction where you want the application to halt program execution. The U sets a breakpoint that stops *before* the instruction is executed. The) sets a breakpoint that stops *after* the instruction is executed.

Let us see how to set a "before" breakpoint at the following instruction:

CP TASKNUM,=P'1'

- Type **U** to the left of the instruction CP TASKNUM,=P'1'. Press PF5 RESUME to continue execution from where the program is currently stopped at the AP instruction.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= ASMDEMO Option # Stmt # Displacement= Margin= 01
Search=
-----
----- R13.TASKNUM | 00000.
-----
++-----
Loc Object Code Addr1 Addr2 Stmt Source Statement
_ 000192 4050 D1B0 001B0 1456 STH R5,MSGLEN
A 000196 FA20 D189 4AE8 00189 02AE8 ==> AP TASKNUM,=P'1'
_ 00019C F920 D189 4AE9 00189 02AE9 1458 CP TASKNUM,=P'2'
_ 0001A2 4720 2328 00328 1459 BH ENDMSG
u 0001A6 F920 D189 4AE8 00189 02AE8 1460 CP TASKNUM,=P'1'
_ 0001AC 4780 227E 0027E 1461 BE SENDSCR4
_ 0001B0 47F0 2328 00328 1462 B ENDMSG
_ 0001B4 1464 RETURN DS 0H
_ 1465 * EXEC CICS SEND FROM(OUTMSG
1466 DFHECALL =X'04043000080000
0001B4 4110 D068 00068 1482 * EXEC CICS SEND FROM(SCREEN
1483 DFHECALL =X'04043000080000
0001DA 4110 D068 00068 1499 * EXEC CICS RETURN
1500 * TRANSID(EIBTRNID)
    
```

The application resumes program execution and continues until it reaches the breakpoint you just set. The program is halted before the breakpoint instruction executes and the following screen appears.

```

CA InterTest for CICS - PROTSYM FILE UNCOND BEFORE BREAKPOINT
COMMAND ==>
Program= ASMDEMO Option # Stmt # Displacement= Margin= 01
Search=
-----
----- R13.TASKNUM | +00001.
-----
++-----
Loc Object Code Addr1 Addr2 Stmt Source Statement
_ 0001A2 4720 2328 00328 1459 BH ENDMSG
U 0001A6 F920 D189 4AE8 00189 02AE8 ==> CP TASKNUM,=P'1'
_ 0001AC 4780 227E 0027E 1461 BE SENDSCR4
_ 0001B0 47F0 2328 00328 1462 B ENDMSG
_ 0001B4 1464 RETURN DS 0H
_ 1465 * EXEC CICS SEND FROM(OUTMSG
1466 DFHECALL =X'04043000080000
    
```

```

0001B4 4110 D068          00068          ), (FB_2,OUTLEN)
                          1482 *          EXEC CICS SEND FROM(SCREEN
                          1483          DFHECALL =X'04043000080000
0001DA 4110 D068          00068          2), (FB_2,MSGLEN)
                          1499 *          EXEC CICS RETURN
                          1500 *          TRANSID(EIBTRNID)
                          1501 *          COMMAREA(TSAREA)
                          1502 *          LENGTH(COMLEN)
                          1503          DFHECALL =X'0E08E00008000
    
```

Notice the U to the left of the highlighted instruction. It identifies the breakpoint as an unconditional "before" breakpoint. For an "after" breakpoint, the application displays a). F or an automatic, conditional, variable-change, or request breakpoint, the application displays an A, C, V, or R, respectively.

 **Note:** The value of TASKNUM is now 1 because the AP instruction executed successfully.

When you are stopped at a breakpoint, you can:

- Scroll or search through your source listing
- Examine and modify main and auxiliary storage
- Add a variable to the Keep window using the K line command
- Set and remove breakpoints using simple line commands such as U,), and X
- Abend your task (with or without a dump)
- Go around a problem by resuming program execution from another location (Type G next to the instruction where you want to resume.)

When debugging your own programs, you typically perform one or more of these activities. ASMDEMO does not have any more errors, so we will complete the demo session.

Remove the Keep Window and Breakpoints

As you continue testing and debugging, you should clean up by removing any breakpoints you no longer need, so that when retesting the program, it will not stop unnecessarily. Now remove the unconditional breakpoint you just set, and also remove TASKNUM from the Keep window.

1. Type **X** to the left of TASKNUM in the Keep window to remove it from the window.
2. Overtyping **U** with **X** to remove the breakpoint.
3. Press Enter.

```

CA InterTest for CICS - PROTSYM FILE UNCOND BEFORE BREAKPOINT
COMMAND ===>
Program= ASMDEMO Option #      Stmt #      Displacement=      Margin= 01
                          Search=
-----
    
```

```
x_____ R13.TASKNUM | +00001.
-----+-----
   Loc  Object Code  Addr1 Addr2 Stmt      Source Statement
- 0001A2 4720 2328          00328 1459      BH  ENDMSG
x 0001A6 F920 D189 4AE8 00189 02AE8 ==>      CP  TASKNUM,=P'1'
- 0001AC 4780 227E          0027E 1461      BE  SENDSCR4
- 0001B0 47F0 2328          00328 1462      B   ENDMSG
- 0001B4          1464 RETURN DS   0H
          1465 *      EXEC CICS SEND FROM(OUTMSG
          1466      DFHECALL =X'04043000080000
                   ),(FB_2,OUTLEN)
                   1482 *      EXEC CICS SEND FROM(SCREEN
                   1483      DFHECALL =X'04043000080000
                   2),(FB_2,MSGLN)
0001B4 4110 D068          00068
                   1499 *      EXEC CICS RETURN
                   1500 *      TRANSID(EIBTRNID)
                   1501 *      COMMAREA(TSAREA)
0001DA 4110 D068          00068
                   1502 *      LENGTH(COMLEN)
                   1503      DFHECALL =X'0E08E000080000
```

The application removes the Keep window and unconditional breakpoint, as shown in the following screen.

```
CA InterTest for CICS - PROTSYM FILE UNCOND BEFORE BREAKPOINT
COMMAND ==>
Program= ASMDEMO Option # Stmt # Displacement= Margin= 01
Search=
-----+-----
   Loc  Object Code  Addr1 Addr2 Stmt      Source Statement
- 0001A2 4720 2328          00328 1459      BH  ENDMSG
- 0001A6 F920 D189 4AE8 00189 02AE8 ==>      CP  TASKNUM,=P'1'
- 0001AC 4780 227E          0027E 1461      BE  SENDSCR4
- 0001B0 47F0 2328          00328 1462      B   ENDMSG
- 0001B4          1464 RETURN DS   0H
          1465 *      EXEC CICS SEND FROM(OUTMSG
          1466      DFHECALL =X'04043000080000
                   ),(FB_2,OUTLEN)
                   1482 *      EXEC CICS SEND FROM(SCREEN
                   1483      DFHECALL =X'04043000080000
                   2),(FB_2,MSGLN)
0001DA 4110 D068          00068
                   1499 *      EXEC CICS RETURN
                   1500 *      TRANSID(EIBTRNID)
                   1501 *      COMMAREA(TSAREA)
                   1502 *      LENGTH(COMLEN)
                   1503      DFHECALL =X'0E08E000080000
000200 4110 D068          00068
                   1518 SENDSCR3 DS   0H
- 000226
```



Note: If there were other variables in the Keep window, they would remain in the window after you removed TASKNUM.

Remove Breakpoints from the Status Display

You can also remove a breakpoint from the Status display. Use the following steps to set another "before" breakpoint, access the Monitoring Status display, and quickly remove the breakpoint from the Status display.

1. Type **U** in front of the RETURN statement and press Enter.
The application sets the unconditional breakpoint at the RETURN.
2. Type **STATUS** on the command line, and press Enter.

```

CA InterTest for CICS - PROTSYM FILE UNCOND BEFORE BREAKPOINT
COMMAND ==>
Program= ASMDEMO Option # Stmt # Displacement= Margin= 01
Search=
-----+-----
Loc Object Code Addr1 Addr2 Stmt Source Statement
_ 0001A2 4720 2328 00328 1459 BH ENDMSG
_ 0001A6 F920 D189 4AE8 00189 02AE8 ==> CP TASKNUM,=P'1'
_ 0001AC 4780 227E 0027E 1461 BE SENDSCR4
_ 0001B0 47F0 2328 00328 1462 B ENDMSG
u 0001B4 1464 RETURN DS 0H
1465 * EXEC CICS SEND FROM(OUTMSG
1466 DFHECALL =X'04043000080000
), (FB_2,OUTLEN)
0001B4 4110 D068 00068 1482 * EXEC CICS SEND FROM(SCREEN
1483 DFHECALL =X'04043000080000
2), (FB_2,MSGLEN)
0001DA 4110 D068 00068 1499 * EXEC CICS RETURN
1500 * TRANSID(EIBTRNID)
1501 * COMMAREA(TSAREA)
1502 * LENGTH(COMLEN)
1503 DFHECALL =X'0E08E000080000
000200 4110 D068 00068 EA), (FB_2,COMLEN)
_ 000226 1518 SENDSCR3 DS 0H

```

The Monitoring Status screen displays. The Monitoring Status display lists all breakpoints and monitoring options for your program. You can easily remove one or all breakpoints without having to locate them in your listing or remember where you set them.

```

----- CA InterTest for CICS MONITORING STATUS -----
COMMAND ==>

Type + to expand or - collapse option levels displayed below,
or R to remove option(s).

Option Description Attributes
_ - ASMDEMO Program monitor entry Assembler
| | Waiting at breakpoint Task 00035, UBP since 10:00 a.m.
_ - |-.ANY User monitoring options Active
| | Symbolic listing file PROTSYM
r - | -UBP Unconditional breakpoint 'RETURN'
| | Option ID D7B97FBE
| | From, to terminals G001, G001
_ - | -SLB Source listing breakpoints G001
| | *** End of data ***

PF1 Help 2 Refresh 3 End 4 Return 5 Collapse 6 Expand
PF7 Backward 8 Forward 9 10 11 12

```

3. Type an **r** to the left of the unconditional breakpoint (UBP).
4. Press Enter.
An asterisk appears next to the UBP breakpoint option, and the Attributes column reads *****REMOVED*****.
5. Press PF2 to refresh the screen.
The unconditional breakpoint entry (UBP) disappears from the status display.
6. Press PF3 to return to the Source Listing Breakpoint screen.

Resume Program Execution

Follow these steps:

1. Press PF5 to continue program execution.

The application resumes program execution. ASMDemo displays the following screen. This screen reminds you that we have touched on just a few of the application's powerful, yet easy-to-use, testing and debugging facilities.

```
*****
****                                     ****
****                               CA InterTest Demo Session          ****
****                                     ****
*****
```

You have completed the sample CA InterTest test session. As part of this session, you:

- * displayed program source code and compiler output online
- * displayed and modified main storage
- * controlled program execution

This is just a fraction of CA InterTest's capabilities. You can also:

- * display or modify any CICS file, or DL/1, DB2, or SQL/DS database
- * set and remove many kinds of monitoring options

Press ENTER or CLEAR to complete the termination.

2. To complete this part of the sample test session, press Clear or Enter to clear your screen.

This demo session taught you the basics of using the application to test and debug an assembler program.

You learned how to perform the following tasks:

- **Select a source program for display**
Go to ITST Menu Option 1.1. Select a program from the list.
- **Set monitoring for a program**
Use the MONITOR command or PF5 from Source Listing.
- **View a Monitoring Status display**
Use the STATUS command or PF12 from Source Listing.
- **Interpret the information the application provides when it detects an error**
View the Automatic Breakpoint and press PF1 for help.
- **Display the value of a data item in a Keep window**
Type K to the left of a variable, place the cursor under the variable and press Enter.
- **Modify main storage**
Overtyping the value in the Keep Window or type D to the left of a variable. Place the cursor under the variable and press Enter for a CORE Main Storage display. On the CORE screen, overtype the values and press Enter.
- **Set an unconditional "before" breakpoint**
Type U to the left of an instruction and press Enter.

- **Remove a breakpoint from the Monitoring Status display**
Use PF12 to view Monitoring Status. Type R next to a UBP and press Enter.
- **Remove a breakpoint from the Source Listing Display**
On Source Listing, overwrite U with X and press Enter.
- **Remove a data item from a Keep window**
Type X to the left of a Keep window entry and press Enter.
- **Resume program execution**
Use the RESUME command or PF5.

Assembler Debugging Programs

This article provides an overview of key functions. For complete information, see [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging) and the Help facility.

- [Checklist of Basic Debugging Tasks \(see page 29\)](#)
- [Display Your Source Listing \(see page 30\)](#)
- [Start a Test Session \(see page 31\)](#)
- [Set and Remove Breakpoints \(see page 32\)](#)
- [Inspect and Modify Main Storage \(see page 34\)](#)
- [Keep Variables in the Keep Window \(see page 36\)](#)
- [Inspect and Modify Auxiliary Storage \(see page 40\)](#)
- [Resume Program Execution \(see page 43\)](#)
- [Abend a Task \(see page 45\)](#)
- [Get Help \(see page 45\)](#)
- [End a Test Session \(see page 46\)](#)

Checklist of Basic Debugging Tasks

Following is a list of debugging tasks:

- Display the source listing
- Set monitoring for the program
- Set breakpoints
- Set other monitoring options as needed
- Initiate the program
- End the test session

When the program is stopped at a breakpoint, you can perform the following tasks:

- Inspect and modify main storage

- Inspect and modify auxiliary storage
- Set and remove breakpoints
- Keep variables in the Keep window to observe changes in their values
- Resume execution
- Abend the task

Display Your Source Listing

You can display your source listing online at any time from CICS using the ITST menus (Option 1 Source). Use PF7 to scroll backward through the listing; use PF8 to scroll forward.

Alternatively, enter the following command directly from CICS:

LIST=progrname

- **progrname**
Specifies the name of the program.

Either method permits access to any CA InterTest for CICS function.



Notes:

- The standard CA InterTest for CICS transaction IDs, such as LIST, ITST, CORE, and FILE, might have been changed at your site. If any of the transactions mentioned in this article do not function as described, contact the person who installed CA InterTest for CICS.
- Depending on your defaults, the options and PF keys available to you might not display at the top of the Source Listing Display. If this information is not displayed, you can view it by pressing PF4 to access the Profile screen, and then setting Display window to T.

Display Sections of Your Program

The LOCATE command or option number (OPTS) listed on the header of the Source Listing screen make it easy to display any section of your Assembler program. Enter the LOCATE command on the command line or the number of the section you want to display in the Option # field and press Enter. For example, if you enter L.C1 in the command line or 1 in the Option # field, CA InterTest for CICS displays the first CSECT.

The following table shows what Assembler areas are displayed when you use the LOCATE command or option numbers:

Command	Option #	Assembler Area Displayed
L.C1	1	1st CSECT
L.MC	5	Macro and Copy Source Catalog for high level output

L.XR	6	Cross Reference
L.LI	7	Literals Cross Reference
L.EM	8	Error Messages

Search for Strings and Labels

You also can search for and display a character string by entering a FIND string NEXT/PREV command on the command line or option 9 (Search forward) or 10 (Search backward) in the Option # field and entering the character string in the Search field. For example, the following screen shows you how to search backward for the string *tasknum*:

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==> f tasknum prev
Program= ASMDemo Option # Stmt # Displacement= Margin= 01
Search=
-----+-----
LOC OBJECT CODE ADDR1 ADDR2 STMT SOURCE STATEMENT
_ 000000 90EC D00C 0000C 1177+ STM 14,12,12(13)
1178+*****
1179+* ESTABLISH CODE ADDRE
1180+*****
R:2347 00000 1181+ USING *-4,2,3,4,7
_ 000004 182F 1182+ LR 2,15
_ 000006 1832 1183+ LR 3,2
_ 000008 4A30 2020 00020 1184+ AH 3,DFHEI4K
_ 00000C 1843 1185+ LR 4,3
_ 00000E 4A40 2020 00020 1186+ AH 4,DFHEI4K
_ 000012 1874 1187+ LR 7,4
_ 000014 4A70 2020 00020 1188+ AH 7,DFHEI4K
1189+*****
1190+* OBTAIN DYNAMIC STORA
1191+*****
_ 000018 4510 2022 00022 1192+ BAL 1,*+10
_ 00001C 0357 1193+ DC AL2(DFHEIEND-DFHEIST
_ 00001E 0000 1194+ DC H'0'
    
```

Exit the Source Listing Facility

Press Clear to return to CICS. The application displays the following message:

```
Lister Function Terminated
```

Start a Test Session

This section explains how to start a testing session.

Turn on Monitoring

To test a program, you must instruct CA InterTest for CICS to *monitor* it. When CA InterTest for CICS monitors a program, it detects and prevents errors before they occur, including the following events:

- All storage violations
- All CICS abends
- Any instruction that would cause a program check or other abend

- All illegal or invalid instructions that would cause CICS to crash
- All wild branches
- All violations of CICS standards

The easiest way to turn on monitoring is from the program's source listing. Follow these steps:

1. Display the source listing as described in the preceding section. A sample Source Listing Display is shown in the previous screen.
2. Type MONITOR on the command line or Press PF5 to set monitoring, which remains in effect for the program until specifically removed.
3. Press Clear to return to CICS.



Note: Before pressing Clear to return to CICS, you might want to set breakpoints directly on the source listing. For more information on setting breakpoints, see [Using Breakpoints \(https://docops.ca.com/display/CAITSD11/Using+Breakpoints\)](https://docops.ca.com/display/CAITSD11/Using+Breakpoints).

Execute a Program

Once you turn on monitoring for a program, you are ready to execute it. After you enter the program's transaction ID, one of the following occurs:

- Your program runs to completion. The results might not be correct.
- CA InterTest for CICS stops your program at a breakpoint; either one set by you or one automatically triggered by CA InterTest for CICS because it detected an error.

Set and Remove Breakpoints

When you test a program, it is important to be able to halt program execution at specified locations. A halt in program execution is called a *breakpoint*.

What You Can Do at a Breakpoint

When your program is stopped at a breakpoint, you can perform the following tasks:

- Examine the source listing
- Inspect and modify main storage
- Inspect and modify auxiliary storage
- Set and remove breakpoints
- Keep variables in the Keep window to observe changes in their values
- Set and remove monitoring options

- Resume execution
- Abend the task

Types of Breakpoints

There are six types of breakpoints, as shown following:

- **Automatic**
The program stops because CA InterTest for CICS detected and prevented an error.
- **Unconditional**
The program stops at the location you specify.
- **Conditional**
The program stops at the location you specify if a condition is met. Optionally, conditional breakpoints can be set to stop at any instruction if a condition is met.
- **Variable-change**
The program stops whenever the value of a specified variable has changed.
- **Request**
The program stops at every CICS command or macro, or at certain CICS commands or macros, or at calls to DL/I, DB2, or software.
- **Single-step**
The program stops after executing one or more instructions.

An automatic breakpoint occurs when CA InterTest for CICS detects an error. When a program is stopped at an automatic breakpoint, you can either correct the error or go around it. You can press PF1 for help on what caused the error and how to use CA InterTest for CICS to fix it. You set the other breakpoints.

Set Unconditional Breakpoints on the Source Listing

To set an unconditional breakpoint on the program source listing, perform the following task:

1. Enter **U** or **)** in column 1 next to the offset where you want the breakpoint. The **U** sets a breakpoint that stops *before* the instruction is executed. The **)** sets a breakpoint that stops *after* the instruction is executed.
2. Press Enter.

The following screen shows how to set a "before" breakpoint at the CP instruction at 1A6.

```
CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>
Program= ASMDEMO Option # Stmt # Displacement= Margin= 01
Search=
-----+-----
Loc Object Code Addr1 Addr2 Stmt Source Statement
R:9 00000 1452 USING COMAREA,R9
- 00018E 1454 CONTINUE DS 0H
- 00018E 4150 01D5 001D5 1455 LA R5,S2LEN
- 000192 4050 D1B0 001B0 1456 STH R5,MSGLEN
- 000196 FA20 D189 4AE8 00189 02AE8 1457 AP TASKNUM,=P'1'
```

```

_ 00019C F920 D189 4AE9 00189 02AE9 1458      CP   TASKNUM,=P'2'
_ 0001A2 4720 2328          00328 1459      BH   ENDMSG
u 0001A6 F920 D189 4AE8 00189 02AE8 1460      CP   TASKNUM,=P'1'
_ 0001AC 4780 227E          0027E 1461      BE   SENDSCR4
_ 0001B0 47F0 2328          00328 1462      B    ENDMSG
_ 0001B4          1464 RETURN DS   0H
          1465 *      EXEC CICS SEND FROM(OUTMSG
          1466      DFHECALL =X'04043000080000
                   ),(FB_2,OUTLEN)
0001B4 4110 D068          00068      1482 *      EXEC CICS SEND FROM(SCREEN
                   1483      DFHECALL =X'04043000080000
                   2),(FB_2,MSGLEN)
0001DA 4110 D068          00068      1499 *      EXEC CICS RETURN

```

After you set a breakpoint, CA InterTest for CICS redisplay the screen with an uppercase U or a). This character remains until you remove the breakpoint.

When you execute the program, CA InterTest for CICS stops the program at the unconditional breakpoint.

You can set breakpoints at any time -- before you execute the program and when the program is stopped. Although where you decide to set breakpoints depends on the specifics of your program, you may want to set breakpoints at the following points:

- At the beginning of the program, so when the program executes you can set additional breakpoints
- At labels, so you can examine the contents of variables at the start of sections
- Before a CALL, so you can dynamically control the program path
- At each location named in an EXEC CICS HANDLE CONDITION, so you can verify error handling

To remove an unconditional breakpoint, type an X over the U or) and press Enter, or type R next to the breakpoint entry on the Monitoring Status display (PF12 from any Source display). You want to remove breakpoints no longer needed so that when you retest the program, you are not stopped unnecessarily.

Inspect and Modify Main Storage

The ability to inspect main storage at a breakpoint means you can determine the values of variables without using a dump. It is easy to find logic errors because you can display the value of a variable where it is defined and at any point in the code where it's referenced. You can also dynamically change its value to correct errors or test other program paths.

Remember that changes to program storage are dynamic; that is, the change affects only the current test session. To correct program errors, you must change the source code and recompile the program.

There are many ways to inspect and modify main storage. We are going to discuss one of the easiest methods of viewing and changing program storage here. For another easy way to display and modify the values of variables, see [Keep Variables in the Keep Window \(see page 36\)](#).

Remember, you can inspect system storage at any time, independent of program monitoring and execution.

Display the Value of a Variable

When you are stopped at a breakpoint, you can request the display of the value of a variable directly from the source listing.

- To display its value where it is defined, type D to the left of the instruction defining it and press Enter, as shown in the next screen.
- To display the value of a variable where it is referenced, type D to the left of the instruction referencing it, place the cursor under any character in the variable, and press Enter.

The following screen shows how to display the value of a variable where it is defined.

```
CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ===>
Program= ASMDEMO Option # Stmt # Displacement= Margin= 01
Search=
-----+-----
Loc Object Code Addr1 Addr2 Stmt Source Statement
_ 000188 366 TSSWITCH DS CL1
d 000189 367 TASKNUM DS PL3
_ 00018C 368 TSTEXT DS CL32
_ 0001AC 369 MSGNAME DS F
_ 0001B0 370 MSGLEN DS H
_ 0001B2 371 RECRBA DS CL4'
_ 0001B8 372 RECPONT DS F
_ 373 *****
_ 0001BC 374 COPY IN25AMP
375= DS 0H
001BC 376=DMAP04AS EQU *
_ 0001BC 377= DS 12C . TI
_ 0001C8 379=RECOU1L DS CL2 . INPUT DATA FI
_ 0001CA 380=RECOU1F DS 0C . DATA FIELD FL
_ 0001CA 381=RECOU1A DS C . DATA FIELD AT
_ 0001CB 382=RECOU1I DS 0CL40 . INPUT DATA
_ 0001CB 383=RECOU10 DS CL40 . OUTPUT DATA
_ 0001F3 385=RECOU2L DS CL2 . INPUT DATA FI
```

CA InterTest for CICS responds by requesting the base register. Specify the register, as shown in the following screen.

```
CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ===>
Program= ASMDEMO Option # Stmt # Displacement= Margin= 01
Search=
-----+-----
```

BASE _____ for TASKNUM

Overtyping Underscores with a Base register For example: R1 or TIOABAR

CA InterTest for CICS then displays the contents of the variable, as shown in the following panel:

```
CA InterTest for CICS - MAIN STORAGE UTILITY - Termid = U015

Starting at Address = 100CE8 Disp Structure Display Format
TASKNUM 189 | ?00000. |
TSTEXT 18C | ..... |
| ..... |
MSGNAME 1AC | 0000000000. |
MSGLEN 1B0 | +00469. |
RECRBA 1B2 | ..... |
RECPONT 1B8 | 0000000000. |
1BC | 000000. |
```

```

DMAP04AS          1BC | .....
                  1BC | .....
RECOUT1L          1C8 | ..
RECOUT1F          1CA | .
RECOUT1A          1CA | .
RECOUT1I          1CB | .
RECOUT10          1CB | .....
-----
PF1 Help      2      3 End      4 Return   5      6 Dump
PF7 Backward  8 Forward  9 Caps Off 10      11 Redisplay 12 Structure
CORE='R13.TASKNUM'
CAIN4713 R13 not found - have assumed its register value
    
```

Modify the Value of a Variable

You can modify the value of a variable by overtyping the bytes in the main storage display and pressing Enter. For example, you could change the contents of TASKNUM in the previous screen by overtyping the question mark with +0 and hit EOF to change the value from binary zeros to a packed decimal value.

Press Clear or PF3 to return to the Source Listing.

Keep Variables in the Keep Window

The Keep window lets you display the values of an unlimited number of variables directly on the source listing. If more than six variables are selected for the Keep window, PF keys 19 and 20 are available to scroll forward and backward through the Keep window. Keeping variables in the window lets you observe changes in their values as the program executes. You also can change values by overtyping the displayed bytes.

When the Keep window is active, the options and PF key functions are not displayed. Press PF4 if you need to refer to them. When all the variables in the Keep window are removed, the options and PF key functions are redisplayed. A command line is available to help you perform CA InterTest for CICS functions when the Keep window is active.

Add a Variable to the Keep Window

You can add variables to the Keep window whenever your source listing is displayed. Identify the variables whose values you want to observe either before you execute the program, or when you are at a breakpoint.

Follow these rules to add a variable to the Keep window:

- To add a variable where it is defined, type K to the left of the statement defining it and press Enter. You will be asked to specify the base register, as shown in the previous screen.
- To add a variable where it is referenced, type K to the left of the statement referencing it, place the cursor under any character in the variable, and press Enter.

When you add a variable to the Keep window, CA InterTest for CICS displays the variable in its data-type appropriate displayable form:

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>>
Program= ASMDEMO  Option #      Stmt #      Displacement=      Margin= 01
                  Search=
-----
    
```

```

----- R13.TASKNUM | 00000.
-----++-----
   Loc Object Code  Addr1 Addr2 Stmt      Source Statement
- 000192 4050 D1B0      001B0 1456      STH  R5,MSGLEN
A 000196 FA20 D189 4AE8 00189 02AE8 ==>      AP  TASKNUM,=P'1'
- 00019C F920 D189 4AE9 00189 02AE9 1458      CP  TASKNUM,=P'2'
- 0001A2 4720 2328      00328 1459      BH  ENDMSG
- 0001A6 F920 D189 4AE8 00189 02AE8 1460      CP  TASKNUM,=P'1'
- 0001AC 4780 227E      0027E 1461      BE  SENDSCR4
- 0001B0 47F0 2328      00328 1462      B   ENDMSG
- 0001B4      1464 RETURN DS   0H
-      1465 *      EXEC CICS SEND FROM(OUTMSG
-      1466      DFHECALL =X'04043000080000
-      1482 *      EXEC CICS SEND FROM(SCREEN
-      1483      DFHECALL =X'04043000080000
-      1499 *      EXEC CICS RETURN
-      1500 *      TRANSID(EIBTRNID)
0001B4 4110 D068      00068
0001DA 4110 D068      00068

```

Display the Variable Structure in a Keep Window

To display the entire DSECT, type D to the left of the variable in the window and press Enter, as shown next.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= ASMDemo Option #      Stmt #      Displacement=      Margin= 01
                          Search=
-----
d_____ R13.TASKNUM | 00000.
-----++-----
   Loc Object Code  Addr1 Addr2 Stmt      Source Statement
- 000192 4050 D1B0      001B0 1456      STH  R5,MSGLEN
A 000196 FA20 D189 4AE8 00189 02AE8 ==>      AP  TASKNUM,=P'1'
- 00019C F920 D189 4AE9 00189 02AE9 1458      CP  TASKNUM,=P'2'
- 0001A2 4720 2328      00328 1459      BH  ENDMSG
- 0001A6 F920 D189 4AE8 00189 02AE8 1460      CP  TASKNUM,=P'1'
- 0001AC 4780 227E      0027E 1461      BE  SENDSCR4
- 0001B0 47F0 2328      00328 1462      B   ENDMSG
- 0001B4      1464 RETURN DS   0H
-      1465 *      EXEC CICS SEND FROM(OUTMSG
-      1466      DFHECALL =X'04043000080000
-      1482 *      EXEC CICS SEND FROM(SCREEN
-      1483      DFHECALL =X'04043000080000
-      1499 *      EXEC CICS RETURN
-      1500 *      TRANSID(EIBTRNID)
0001B4 4110 D068      00068
0001DA 4110 D068      00068

```

CA InterTest for CICS responds by displaying the variable and all the items below it in the same DSECT.

Modify the Value of a Variable in a Keep Window

Modify the value of a variable in the Keep window by overtyping the displayed bytes, as you would overtype the bytes in a main storage display. For more information, see [Inspect and Modify Main Storage \(see page 34\)](#).



Note: If you try to change password protected storage areas not owned by your program, CA InterTest for CICS prompts you to enter the password.

Remove a Variable from the Keep Window

To remove a variable from the Keep window, type X to the left of the variable, as shown next. In this example, three variables will be removed from the Keep window: MSGLEN, TSQTRAN, TSSWITCH, and TSQTERM.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= ASMDEMO Option # Stmt # Displacement= Margin= 01
Search=
-----
----- R13.TASKNUM | 00000.
x----- R13.MSGLEN | +00469.
x----- R13.TSQTRAN | DEMA
x----- R13.TSSWITCH |
x----- R13.TSQTERM | U015
-----
++-----
Loc Object Code Addr1 Addr2 Stmt Source Statement
- 000180 362 TSQNAME DS 0CL8
- 000180 363 TSQTRAN DS CL4
- 000184 364 TSQTERM DS CL4
- 000188 365 TSAREA DS 0CL36
- 000188 366 TSSWITCH DS CL1
- 000189 367 TASKNUM DS PL3
- 00018C 368 TSTEXT DS CL32
- 0001AC 369 MSGNAME DS F
- 0001B0 370 MSGLEN DS H
- 0001B2 371 RECRBA DS CL4'
- 0001B8 372 RECPONT DS F
- 373 *****
    
```

When all variables have been removed, the Keep window is replaced by the options and PF key functions.

Use Variable-Change Breakpoints to Detect Changing Values

You can request that the application halt execution whenever the value of a specified variable changes. This is called a variable-change breakpoint. If you set a variable-change breakpoint while the variable is also listed in a Keep window, when the breakpoint is triggered, you will immediately see the variable's new value at the top of the breakpoint display.

Set a Variable-Change Breakpoint

A quick way to set a variable-change breakpoint is from a Keep window listing of the variable. Type V to the left of the variable in the Keep window, as shown following, and press Enter.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= ASMDEMO Option # Stmt # Displacement= Margin= 01
Search=
-----
v_____ R13.TASKNUM | 00000.
-----
++-----
Loc Object Code Addr1 Addr2 Stmt Source Statement
- 000192 4050 D1B0 001B0 1456 STH R5,MSGLEN
- A 000196 FA20 D189 4AE8 00189 02AE8 ==> AP TASKNUM,=P'1'
- 00019C F920 D189 4AE9 00189 02AE9 1458 CP TASKNUM,=P'2'
- 0001A2 4720 2328 00328 1459 BH ENDMSG
- 0001A6 F920 D189 4AE8 00189 02AE8 1460 CP TASKNUM,=P'1'
- 0001AC 4780 227E 0027E 1461 BE SENDSCR4
- 0001B0 47F0 2328 00328 1462 B ENDMSG
- 0001B4 1464 RETURN DS 0H
- 1465 * EXEC CICS SEND FROM(OUTMSG
- 1466 DFHECALL =X'04043000080000
    
```

```

0001B4 4110 D068          00068          ), (FB_2,OUTLEN)
                                1482 *          EXEC CICS SEND FROM(SCREEN
                                1483          DFHECALL =X'04043000080000
0001DA 4110 D068          00068          2), (FB_2,MSGLEN)
                                1499 *          EXEC CICS RETURN
                                1500 *          TRANSID(EIBTRNID)

```

In this example, CA InterTest for CICS sets a variable-change breakpoint for TASKNUM. This means CA InterTest for CICS halts execution whenever the value of TASKNUM changes from its current value, as displayed in the Keep window.

You can also set variable-change breakpoints from various parts of your program listing using the letter V as follows:

- To set a variable-change breakpoint from where a variable is defined, type V to the left of the statement defining it, and press Enter.
- To set a variable-change breakpoint from where a variable is referenced, type V to the left of the statement referencing it, place the cursor under any character in the variable, and press Enter.

You can set a variable-change breakpoint any time during execution, or from a listing of your program before execution begins. If you set the breakpoint before execution begins, the initial value of the variable is its value when it first becomes known to CA InterTest for CICS during execution. Any change from the initial value triggers the variable-change breakpoint.

View the Variable-Change Breakpoint

The uppercase V that appears to the left of the highlighted statement easily identifies a breakpoint display as a variable-change breakpoint, as shown in the following panel:

```

CA InterTest for CICS - PROTSYM FILE COND BEFORE BREAKPOINT
COMMAND ==>>
Program= ASMDEMO Option # Stmt # Displacement= Margin= 01
Search=
-----
----- R13.TASKNUM | +00001.
-----
++-----
Loc Object Code Addr1 Addr2 Stmt Source Statement
000196 FA20 D189 4AE8 00189 02AE8 1457 AP TASKNUM,=P'1'
V 00019C F920 D189 4AE9 00189 02AE9 ==> CP TASKNUM,=P'2'
==>
==> CAIN3630 Conditional breakpoint requested at offset .ANY
==> IF='R13.TASKNUM'.NE.'R13.TASKNUM'
==>
- 0001A2 4720 2328 00328 1459 BH ENDMSG
- 0001A6 F920 D189 4AE8 00189 02AE8 1460 CP TASKNUM,=P'1'
- 0001AC 4780 227E 0027E 1461 BE SENDSCR4
- 0001B0 47F0 2328 00328 1462 B ENDMSG
- 0001B4 1464 RETURN DS 0H
1465 * EXEC CICS SEND FROM(OUTMSG
1466 DFHECALL =X'04043000080000
0001B4 4110 D068 00068 ), (FB_2,OUTLEN)
1482 * EXEC CICS SEND FROM(SCREEN
1483 DFHECALL =X'04043000080000

```

The lines below the highlighted line further identify the breakpoint. If you set more than one variable-change breakpoint, this information lets you know which variable triggered the breakpoint.

Why does the breakpoint display refer to this as a conditional breakpoint? Because a variable-change breakpoint is actually a special type of conditional breakpoint. The condition CA InterTest for CICS checks for, at ANY statement, is shown as:

The ability to inspect and modify auxiliary storage when a program is stopped at a breakpoint lets you change test data in the middle of a test session. Any changes you make are permanent; that is, changes to the file remain after the test session ends.

Inspect Auxiliary Storage

```
----- CA InterTest for CICS MONITORING STATUS -----
OPTION ==> 1

Select an auxiliary storage type, specifying optional criteria below.

1 Files           - Display/select files for access
2 DB2 database    - Invoke DB2 SQL interface facility
3 DL/I database   - Access DL/I database
4 TD queues       - Display/select transient data queues for access
5 TS queues       - Display/select temporary storage queues for access

Type specific or generic file/queue name(s):

filename _____
```

Alternatively, if you are at a clear CICS screen, you can invoke the FILE transaction. CA InterTest for CICS displays the following panel:

```
DATATYPE= FC FILEID=          MODE=      LOG=OFF  TODEST=      PASSWORD=
FUNC=      SUBFUNC=          RETMETH=  ARGTY=      SRCHTYP=
MESSAGE=
RETNRCID=
  RCID=
  DATA=
FORMAT= D 00112233 44556677 8899AABB CCDDEEFF *0123456789ABCDEF*
LOC 0000 .....
```

CA InterTest for CICS V10.0
Copyright © 2016 CA. All rights reserved.

```
-----
1 Help      2 Format C  3 End      4 BEGB      5          6 DataType DL
7 Page bwd  8 Page fwd  9 Caps Off 10 Top      11 Bottom   12
```



Note: If the FILE transaction does not display the previous screen, it is probably because the person who installed CA InterTest for CICS changed the name of the FILE transaction. Contact that person to find out the correct name.

After you enter the file or database and, optionally, the record or segment, CA InterTest for CICS displays the data. A sample VSAM record in dump format follows.

```
DATATYPE= FC FILEID= INVNTRY  MODE=      LOG=OFF  TODEST=      PASSWORD=
FUNC=      SUBFUNC=          RETMETH=  ARGTY=      SRCHTYP=
MESSAGE= RECORD OBTAINED FOR VIEWING
RETNRCID=F0F0F0F0F0F0F0F5
  RCID=C'000000005'
  DATA=
FORMAT= D 00112233 44556677 8899AABB CCDDEEFF *0123456789ABCDEF*
LOC 0000 .....
```

0000	F0F0F0F0	F0F0F0F0	F0F54040	0087338F	0000000005	DSORG=VSKS
0010	F0F3F9F9	0407835C	40404040	40404040	0399...	RECFM=VB
0020	40404040	40404040	C1C2C340	C3D6D9D7	ABC CORP	LRECL=0050
0030	F0F9F1F2	F8F74040	0000598C	40404040	091287	BLKSIZE=0000
0040	D1D6C3D5	404040E2	D4C9E3C8	40404040	JOHN SMITH	KEYPOS=0000
0050						KEYLEN=0A

TSQTRAN	B8	00000000	
TSQTERM	BC	00000000	
TSAREA	C0	00C3D6D9 D7F6F0F5 F4F40000 000000C7		.CORP60544.....
G				
		D6D6C440 D7D9C9C3 C5000000 00000000		00D PRICE.....
.				
		000000C2		...B
TSSWITCH	C0	00		.
TASKNUM	C1	C3D6D9		COR
TSTEXT	C4	D7F6F0F5 F4F40000 000000C7 D6D6C440		P60544.....GOOD
		D7D9C9C3 C5000000 00000000 000000C2		PRICE.....
B				

1 Help	2 Format D	3 End	4 BEGB	5 PREV
6 Data Type DL				
7 Page bwd	8 Page fwd	9 Caps Off	10 Top	11 Bottom
				12

Structured format lists the contents of each data name in both hexadecimal and character format. To modify the record or segment, overwrite either the hexadecimal or character data and press Enter. To rewrite the modified record, type PUT in the FUNC= field and press Enter.

When a program is stopped at a breakpoint, you can substitute an asterisk (*) for the symbolic-name. For example, if program PAYROLL is stopped at a breakpoint and you enter:

```
USE=* .payrec
```

CA InterTest for CICS uses the program named PAYROLL and the PAYREC structure.

To begin the structure with a specific data name, enter this command in the DATA field:

```
FIND=data-name
```

Where *data-name* is the data name to be displayed.

Resume Program Execution

When you finish performing breakpoint activities such as displaying and modifying main or auxiliary storage, you can do the following actions:

- Resume program execution from a specific location
- Single-step through execution one instruction at a time

Resume Execution from a Specific Location

To resume program execution from the breakpoint location where your program is stopped, the source listing screen must be displayed. Press Clear from any CA InterTest for CICS screen until the breakpoint screen reappears.

Enter the GO command on the command line or Press PF5 to continue from the current breakpoint. Execution continues until the program is halted at another breakpoint or the task completes. If you do not want to halt the program at any of your preset breakpoints, you can enter the RUN command on the command line and the program will continue execution from the current breakpoint and preserve all preset breakpoints.

If you are stopped at an automatic breakpoint, you must first correct the error that triggered the breakpoint before continuing execution from that point. Or, you can go around the error and continue execution from another location.

To resume program execution from another location, follow these steps:

1. Display the instruction where you want to resume execution. For more information, see [Resume and Execution Options \(https://docops.ca.com/display/CAITSD11/Resume+and+Execution+Options\)](https://docops.ca.com/display/CAITSD11/Resume+and+Execution+Options).
2. Type G to the left of that instruction.
3. Press Enter.

or

- Enter GO label or *GO linenum*ber on the command line and press enter

Program execution resumes from that instruction label or line number.

The following example shows how to resume execution from a location prior to an automatic breakpoint. This is useful if you corrected a record or dynamically modified an incorrect file name and now want the program to reread the record or file.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING BREAKPOINT
COMMAND ==>
Program= ASMDEMO  Option #      Stmt #      Displacement=      Margin= 01
                        Search=
-----+-----
   LOC  OBJECT CODE  ADDR1 ADDR2  STMT      SOURCE STATEMENT
_ 001B94 D201 48FA 4A0C 028FA 02A0C 2099      MVC  FIRST,=CL2'N0'
      2100 *      EXEC  CICS SEND
      2101 *      MAP  ('DMAP04')
      2102 *      MAPSET ('IN25AMP')
      2103 *      ERASE
      2104 *      WAIT
      2105      DFHECALL =X'1804D000080000
Active Usings: ASMDEMO(X'4000'),R2,R3,R4,R8 COMAREA(X'1000'),R9 DFHEIBLK(
001B9A 4110 D068      00068      ),(____RF,DMAP04),,
_ 001BC2 9200 B01A      0001A      2120      MVI  EIBAID,X'00'
      2121 *      EXEC  CICS RECEIVE
_ 001BC6 4110 D068      00068      2122      DFHECALL =X'04020000080000
_ 001BDC 956D B01A      0001A      2132      CLI  EIBAID,DFHCLEAR
_ 001BE0 4780 2300      00300      2133      BE   ENDMSG
g 001BE4 47F0 3FAE      01FAE      2134      B    READATA
_ 001BE8      2136 SENDMP02 DS  0H
      2137 * SEND OUT A SCREEN
_ 001BE8 D201 48FE 49FE 028FE 029FE 2138      MVC  MAPNUM,=CL2'03'
    
```

When you press Enter, execution resumes from offset 1BE4, the location you identified with a G.

Single-Stepping

Single stepping enables your program to execute one Assembler instruction and then stop again. You can adjust the step amount so that the program executes a specified number of instructions between stops.

To single-step: Type NEXT in the command line and press Enter, or press PF10 from the source listing. Your program executes the next instruction and then stops.

To single-step from a different program location, perform the following steps:

1. Display the statement where you want to resume execution. For more information, see [Display Your Source Listing \(see page 30\)](#).

2. Type G to the left of that statement.
3. Type NEXT in the command line and press Enter or press PF10.

To adjust the step amount, perform the following steps:

1. Press PF4 to access the Source Listing Profile screen.
2. Change the number in the Stepping amount field.
3. Press Enter. CA InterTest for CICS changes the step amount and redisplay the source listing. If Titles are set to **on** (an option on the Source Listing Profile screen), you will see the new step amount displayed next to PF10 at the top of the Source Listing Breakpoint screen.

Abend a Task

When your program stops at a breakpoint, you can abend your task rather than resume execution. To abend a task, enter the ABEND command from the breakpoint display and press Enter. CA InterTest for CICS displays the following screen.

```
----- CA InterTest  ABEND BREAKPOINTED TASK  -----
COMMAND ==>
```

Type an abend code, then press ENTER.

Abend Code ____ Abend code options are:

blanks	Normal abend, no dump
XXXX	Abend exits cancelled, no dump
your code	Your abend code, dump taken

- To cancel the abend, press PF3 or Clear. CA InterTest for CICS redisplay the breakpoint screen.
- To abend without a dump:
 - For a normal abend, press Enter
 - To cancel all program exits, specify code XXXX and press Enter
- To abend *with* a dump, specify a four-character dump code and press Enter. The dump code you specify cannot be XXXX and cannot begin with the letter A.

Get Help

Help is always available when you are using CA InterTest for CICS. You can also use Help to initiate CA InterTest for CICS, type in Help on a clear screen under CICS.

Get Help Using CA InterTest for CICS

When you are using CA InterTest for CICS, you can press PF1 at any time for information on CA InterTest for CICS functions. For example, the following screen lists the Help topics for the Source Listing facility.

```
CA INTERTEST - INTERACTIVE HELP FACILITY -
TUTORIAL: SOURCE LISTING FACILITY - FOR ASSEMBLER PROGRAMS
```

05 Set Monitoring	45 Set/Remove other Monitoring Options
10 Locating areas in the listing	50 Displaying data items
20 Search for a Data string	60 Resume TASK execution
30 PF key definitions	65 Abending your task
32 Primary Line Commands	70 PROFILE, AUTOSTEP, AUTOKEEP Options
35 Set the Display Margins	75 Code Coverage Option
40 Set/Remove Breakpoints	80 View Program Backtrace

(end)

 ENTER A SELECTION CODE FROM THE MENU, OR N FOR NEXT PAGE, P FOR PRECEDING PAGE,
 M FOR RETURN TO PREVIOUS MENU, CLEAR TO EXIT, OR MESSAGE NUMBER. ==> 05

Get Help to Correct an Error

Whenever your program stops at an automatic breakpoint, CA InterTest for CICS for CICS displays a short message identifying the reason for the breakpoint. Press PF1 for information on what caused the error that triggered the breakpoint and how to fix it. The following screen explains how to correct an AEIL abend.

CA INTERTEST - INTERACTIVE HELP FACILITY -
 TUTORIAL: ERROR MESSAGE AEIL

The dataset name referred to in the DATASET option cannot be found in the FCT. Your program did not have a Handle Condition for this error. Or a second Handle Condition without a routine for this error superseded the one that had a routine for this error.

WHAT YOU CAN DO: If the named file was entered incorrectly and the one you wanted exists, you may use the Replace File Option to dynamically replace the file name and then use the resume task facilities to execute the CICS request again. To perform the above functions from the Source Listing Breakpoint screen you would:

1. Key =20s in the Option # field and press ENTER.
2. Tab to Replace file name: Key in the incorrect file name.
3. In the next field, key in the correct file name and press ENTER.
4. Press CLEAR to return to your Source Listing Breakpoint screen.
5. Key in G to resume execution at the beginning of the EXEC CICS command and press ENTER.

(continued)

 ENTER N FOR NEXT PAGE, P FOR PRECEDING PAGE, F FOR FIRST PAGE, OR -
 M FOR RETURN TO PREVIOUS MENU, CLEAR TO EXIT, OR MESSAGE NUMBER. ==> N



Note: When you leave and then return to an automatic breakpoint screen, the error message may no longer appear. For help on the cause of the error, press Clear to redisplay the error message before pressing PF1.

End a Test Session

After your program completes execution or you abend your task, you return to CICS.

Status After Testing

The following table shows the status of your program after testing:

Program Part	Any Changes
program source code	not modified
file/database updates by the program	permanent
file/database updates you made	permanent
CA InterTest for CICS monitoring, breakpoints, and other options set for the program	remain in effect

Correct the Source Code

At some point you might want to correct the source code and continue testing from a clean version of the program.

After you modified the source code and recompiled the program, you can load the new copy of the program by using the New Program Copy option on the Program Monitoring menu (ITST 2.1).

The New Program Copy option resets symbolic breakpoints and other monitoring options for the recompiled program. It also resets the PPT entry to the program's new library address.

Assembler Advanced Monitoring Features

This article describes just a few of the more commonly used features. For information about all product features, see [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging) and the Help facility .

This article contains the following topics:

- [Set Options from the Monitoring Menus \(see page 47\)](#)
- [Conditional Breakpoints \(see page 49\)](#)
- [Request Breakpoints \(see page 51\)](#)
- [Backtrace Facility \(see page 52\)](#)
- [Set the Code Counting Coverage Option \(see page 55\)](#)
- [Replacement, Protection, and Special Options \(see page 56\)](#)
- [Composite Module Testing \(see page 58\)](#)
- [Dump Analysis with CA SymDump for CICS \(see page 59\)](#)

Set Options from the Monitoring Menus

The CA InterTest for CICS monitoring menus are the easiest way to set many monitoring options. Access these menus from the Primary Option Menu by selecting Option 2 - Monitoring, shown in the following panel:

```
----- CA InterTest for CICS PRIMARY OPTION MENU -----
OPTION ==> 2
```

- 1 Source - Display/select program source files/listings
 - 2 Monitoring - Display/modify CA InterTest monitoring/activity
 - 3 Main storage - Display/modify CICS storage areas
 - 4 Auxiliary storage - Display/access databases/files/queues
 - 5 Dump analysis - Invoke CA SymDump CICS dump/trace capture facility
 - 6 Product help - Invoke CA InterTest product help facility
 - 7 Status/Maintenance - Product status and maintenance functions
 - 8 What's new? - Display information about CA InterTest for CICS
 - X Exit - Terminate menu processing
- .
- .
- .

CA InterTest for CICS displays the Monitoring Menu, from which you can select a variety of different monitoring options. For a detailed explanation of each of these options, see [Monitoring Menu Options \(https://docops.ca.com/display/CAITSD11/Monitoring+Menu+Options\)](https://docops.ca.com/display/CAITSD11/Monitoring+Menu+Options).

```
----- CA InterTest for CICS MONITORING MENU -----
OPTION ==>

    1 Programs          - Display/modify program monitoring options
    2 Transactions      - Display/modify transaction monitoring options
    3 Terminals         - Display/modify terminal monitoring options
    4 Status            - Display/remove current monitoring options
    5 Active tasks      - Display/reconnect/purge active monitored tasks
    6 System-wide       - Display/modify global system-wide options
    .
    .
    .
```

Option 1 - Programs accesses the Program Monitoring Menu, where you can set and remove any option for a program, and request a status display.

```
----- CA InterTest for CICS PROGRAM MONITORING -----
COMMAND ==>

    Type information and S to set or R to remove option(s) below.

    Program . . asmdemo_      Program name (or .ALL, .OPTIONS or generic)
    User ID . .      _____ User (or .ANY) for whom the program is monitored

    Option      Description                                     More:  +
    - Status    Display and/or remove monitoring options (S only)
    - Monitor   Monitoring (R removes monitoring and all options previously set)
    - UBP       Unconditional breakpoints (specific program only)
    - CBP       Conditional breakpoints (specific program only)
    - RBP       Breakpoints for CICS, DB2, DL/I or external CALL requests
    - Stmt Trace Statement tracing and data monitoring (COBOL only)
    - New copy  Fetch new copy of program and reset monitoring options (S only)
    - Commands  Indirect commands defined for a specific COBOL or PL/1 program
    - Replace   CICS resource name replacement options
    - Protect   Storage protection monitoring options
    - Special   Other options (storage allocation, file updating, etc.)
    - Composite Monitor multi-CSECT program's separately compiled components

    PF1 Help    2          3 End          4 Return    5          6
    PF7 Backward 8 Forward  9          10         11         12
```

Quick access to the Program Monitoring menu is available before or during a session.

The following table contains the fastpath entries that you can use to access the Program Monitoring menu:

From	Fastpath entry
CICS, clear screen	ITST 2.1

From	Fastpath entry
Primary Option Menu, Option field	2.1
Source Listing display, Command line	=2.1
Breakpoint display, Command line	=1.2.1

After processing any program options from this menu, press PF3 to exit back to your test session. If you entered directly from CICS, you exit back to the ITST Primary menu. From a Source Listing or breakpoint, PF3 takes you back to that display.

Conditional Breakpoints

Conditional breakpoints are breakpoints that take effect when a specific condition is met, such as when a counter exceeds a value. These breakpoints can help you isolate complex problems.

Set Conditional Breakpoints

To set a conditional breakpoint, begin by typing **c** to the left of the offset where you want the breakpoint, as shown next, and press Enter.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>
Program= ASMDEMO Option # 9 Stmt # Displacement= Margin= 01
Search= GETCOM
-----+-----
  Loc Object Code Addr1 Addr2 Stmt Source Statement
- 000100 4980 B018 00018 1380 CH R8,EIBCALEN
  000104 4740 218A 0018A 1381 BL GETCOM
- 000108 957D B01A 0001A 1382 CLI EIBAID,DFHENTER
- 00010C 4780 218E 0018E 1383 BE CONTINUE
c 000110 47F0 2142 00142 1384 B SENDSCR1
  000114 0024 1386 TSQLEN DC H'36'
- 000116 0001 1387 TSQITEM DC H'1'
- 000118 1389 WRITETSQ DS 0H
- 000118 9240 D188 00188 1390 MVI TSSWITCH,C' '
  1392 * EXEC CICS WRITEQ TS
  1393 * QUEUE(TSQNAME)
  1394 * FROM(TSAREA)
  1395 * LENGTH(TSQLEN)
  1396 * MAIN
  1397 DFHECALL =X'0A02E000080000
  00011C 4110 D068 00068 A),(FB_2,TSQLEN)
- 000142 1412 SENDSCR1 DS 0H
Active Usings: ASMDEMO,R2,R3,R4,R7 DFHEIBLK,R11 DFHEISTG,R13
    
```

CA InterTest for CICS then displays the menu on which you define the condition. For example, the following screen sets the conditional breakpoint at the offset you identified in the previous screen only if TSSWITCH is equal to 3. The entries are as follows:

```

LEFT SIDE of condition:
      Data Name r13.tsswitch
      OPERATOR: eq
RIGHT SIDE of condition:
      Literal 3

          CA-InterTest MONITORING COMMAND BUILDER - CONDITIONAL BREAKPOINT

Enter LEFT SIDE of condition (select one):
      Data Name r13.tsswitch -----
      Special keywords: _____ Register # __ COBOL BLL # __

Enter OPERATOR (EQ, NE, GT, LT, GE, LE): eq Length: Left ___ Right __
    
```

```

Enter RIGHT SIDE of condition (select one):
  Data Name -----
  Special keywords: ----- Register # __ COBOL BLL # __
  Literal 3 -----

Optional offset: Enter + - @ OR % followed by a value.
  Left side ----- Right side -----

_ ENTER S to Drop monitoring on a true condition

For location:
      000110 47F0 2142          00142 1384          B SENDSCR

  Use Help or documentation for use of special keywords
PF1 Help      2          3 End      4 Return    5          6
PF7           8          9          10         11         12
    
```

To process the condition, press Enter. CA InterTest for CICS generates the command syntax for the conditional breakpoint, and returns you to the Source Listing screen.

Remove Conditional Breakpoints

To remove a conditional breakpoint, type X over the C and press Enter, as shown in the following screen.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>
Program= ASMDEMO Option # 9 Stmt # Displacement= Margin= 01
Search= GETCOM
-----+-----
  Loc Object Code Addr1 Addr2 Stmt Source Statement
- 000100 4980 B018 00018 1380 CH R8,EIBCALEN
- 000104 4740 218A 0018A 1381 BL GETCOM
- 000108 957D B01A 0001A 1382 CLI EIBAID,DFHENTER
- 00010C 4780 218E 0018E 1383 BE CONTINUE
x 000110 47F0 2142 00142 1384 B SENDSCR1
- 000114 0024 1386 TSQLEN DC H'36'
- 000116 0001 1387 TSQITEM DC H'1'
- 000118 1389 WRITETSQ DS 0H
- 000118 9240 D188 00188 1390 MVI TSSWITCH,C' '
  1392 * EXEC CICS WRITEQ TS
  1393 * QUEUE(TSQNAME)
  1394 * FROM(TSAREA)
  1395 * LENGTH(TSQLEN)
  1396 * MAIN
  1397 DFHECALL =X'0A02E000080000
00011C 4110 D068 00068 A),(FB_2,TSQLEN)
- 000142 1412 SENDSCR1 DS 0H
  Active Usings: ASMDEMO,R2,R3,R4,R7 DFHEIBLK,R11 DFHEISTG,R13
    
```

In addition, you can remove the conditional breakpoint from the Status Display, as shown in the following steps:

1. Press PF12 from any Source screen to display the Monitoring Status display.
2. Scroll to the line where the conditional breakpoint (CBP) option is listed for the program.
3. Type R next to the entry and press Enter.
4. Return to the Source screen using PF3 End.

Request Breakpoints

Request breakpoints are breakpoints that take effect prior to CICS commands and macros and other program calls, such as calls to DL/I or DB2. With one specification, you can set breakpoints at all CICS commands or at specific commands, such as all REWRITE commands.

Use request breakpoints to stop a program before the following commands:

- Every CICS command or macro
- Specific types of CICS commands, such as all File Control or Program Control commands
- Specific commands, such as all READ or WRITE commands

Set Request Breakpoints

To set request breakpoints, perform the following steps:

1. Select option 2 - Monitoring on the Primary Option Menu. CA InterTest for CICS displays the Monitoring Menu.
2. Select option 1 - Programs. CA InterTest for CICS displays the Program Monitoring screen.

```

----- CA InterTest PROGRAM MONITORING -----
COMMAND ==>

  Type information and S to set or R to remove option(s) below.

  Program . . asmdemo_      Program name (or .ALL, .OPTIONS or generic)
  User ID . .      _____ User (or .ANY) for whom the program is monitored

  Option      Description                                          More: +
  _ Status    Display and/or remove monitoring options (S only)
  _ Monitor   Monitoring (R removes monitoring and all options previously set)
  _ UBP       Unconditional breakpoints (specific program only)
  _ CBP       Conditional breakpoints (specific program only)
  s RBP       Breakpoints for CICS, DB2, DL/I or external CALL requests
  _ Stmt Trace Statement tracing and data monitoring (COBOL only)
  _ New copy  Fetch new copy of program and reset monitoring options (S only)
  _ Commands  Indirect commands defined for a specific COBOL or PL/1 program
  _ Replace   CICS resource name replacement options
  _ Protect   Storage protection monitoring options
  _ Special   Other options (storage allocation, file updating, etc.)
  _ Composite Monitor multi-CSECT program's separately compiled components

  PF1 Help    2          3 End      4 Return   5          6
  PF7 Backward 8 Forward  9          10         11         12

```

3. In the Program field, type the name of the program for which you want to set request breakpoints.
4. Type S to the left of the RBP (request breakpoints) option. Press Enter.

The Request Breakpoint Selection menu appears on which you specify the type of request breakpoint, such as: all CICS commands, File Control, Task Control, and so on, as shown in the following panel:

```

CA InterTest MONITORING COMMAND BUILDER - REQUEST BREAKPOINT SELECTION 13
Set one or more types of Request Breakpoints in:
PROG=ASMDEMO

```

```

_ ALL commands          _ DL/I    _ DB2    _ CALLs
_ Address, Assign,      _ Storage Control    _ BMS
  Handles, Push, Pop   _ Program Control    _ Trace Control
_ Terminal Control     _ Interval Control    _ Dump Control
_ File Control         _ Task Control       _ Batch Data Interchange
_ TD Control          _ Journal Control    _ Built-In Functions
_ TS Control          _ Syncpoints         _ Sys Prog Functions
_ Web access         _ Business Trans     _ 3270 Bridge

Enter 'n' to stop only every n'th time:
Term ID (or .ANY or .NO) where breakpoints will take effect:  ----
Term ID (or .ANY) that will receive the breakpoints:         ----
User ID (or .ANY) who will execute the program:              -----

PF1 Help      2          3 End      4 Return   5          6
PF7           8          9          10         11         12
    
```

Subsequent menus prompt you for the specific command, such as all File Control commands, READ, WRITE, REWRITE, and so on.

 **Note:** Type X in front of each type of command for which you want to set request breakpoints.

After setting your request breakpoints, you are returned to the Program Monitoring menu. Use PF3 to return to the previous menu or Source display, use PF4 to go to the Primary Option Menu.

Remove Request Breakpoints

To remove request breakpoints, perform the following steps:

1. From the Source Listing display or breakpoint display, type STATUS on the Command line and press Enter. CA InterTest for CICS displays the Monitoring Status, showing all options for the current program.
2. Type R to the left of the request breakpoint (RBP) option you want to remove and press Enter. CA InterTest for CICS places an asterisk to the left of the breakpoint to indicate that the command has been processed.
3. Press PF3 to return to the Source Listing or Breakpoint display.

Backtrace Facility

When your program is at a breakpoint, you will find it useful to examine the backtrace. The backtrace shows the logic of the program and explains how execution reached this point.

To enter the Backtrace facility, press PF11 from the Source Listing Breakpoint screen, or, type BTRACE in the Command line and press Enter. CA InterTest for CICS positions the Backtrace Summary Screen at the last executed statement, which is the current breakpoint location.

To view your program's backtrace, statement-by-statement, press PF2 from the Backtrace Summary screen to access the Source Listing Backtrace.



Note: The execution PF keys are disabled and are replaced with special forward and backward tracing PF keys, while you are using the Backtrace facility.

The following screen shows the Backtrace Summary screen:

```

CA InterTest for CICS - BACKTRACE SUMMARY

Program= ASMDEMO                               From 0001 To 0006 Of 0006

Specify S then ENTER to display Source Listing BACKTRACE
-----
PFKS 1 Help      2 Backtrace 3 End          4          5 1st Stmt   6 Last Stmt
      7 Backward  8 Forward  9          10         11 Prev Bloc 12 Next Bloc
-----
S Bkmk      Stmt Block | Source Listing
- ----      +0 ... +18 | +   STM 14,12,12(13)      SAVE CALLER'S R
- ----      +22 ... +26 | +   L   15,=V(DFHEAI0)
- ----      +28 ... +9E | +   ST 13,DFHEISA-DFHEISTG+4(,1) CHAIN TO
- ----      +A0 ... +CA |
- ----      +CC ... +E4 |      CLI TSSWITCH,DFHCLEAR      CLEAR KEY S
* ----      +166 ... BKPT |      LA  R5,S2LEN                LENGTH OF V
-----

```

CAIN2988 First and Last Backtrace Stmt Block

Read the Backtrace Summary

The Backtrace Summary summarizes the program's execution path using statement blocks. By tracing the execution path, block by block, you can follow the flow of your program's logic. The statement block column is located on the left side of the screen and contains two columns of numbers:

- Statement numbers in the left column received control from a branch, command level CALL, a PERFORM, or a GOTO.
- Statement numbers in the right column issued the branch to the next statement identified.

From the Backtrace Summary screen you can do the following tasks:

- Reposition the display to view statement-by-statement details of the program's execution path by:
 - Pressing PF2
 - Marking a statement block with S, and then pressing Enter
- Assign a *bookmark*, consisting of one to four characters, to one or more backtrace positions for faster and easier navigation through the Backtrace Summary
- Trace program execution using the following PF keys:
 - **PF5 1st Stmt**
Repositions the Backtrace Summary screen to the first or oldest entry in the Backtrace.
 - **PF6 Last Stmt**
Repositions the Backtrace Summary screen to the last or newest entry in the Backtrace.

- **PF7 Backward**
Scrolls the Backtrace Summary screen backward (up) a full screen.
- **PF8 Forward**
Scrolls the Backtrace Summary screen forward (down) a full screen.
- **PF11 Prev Bloc**
Displays the previous statement block.
- **PF12 Next Bloc**
Displays the next statement block.

Access the Source Listing Backtrace

To switch to the Source Listing Backtrace and to view the backtrace statement-by-statement, press PF2, or key S next to a statement block in the Select column of the Backtrace Summary to select a starting position and press Enter.

The following screen shows the Source Listing Backtrace:

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING BACKTRACE
COMMAND ==>
Program= ASMDEMO  Option #      Stmt #      Displacement=      Margin= 01
                          Search=
===== Backtrace at +0016E (00166 -> curr. executed 1 times) <=====
      LOC OBJECT CODE  ADDR1 ADDR2 STMT  SOURCE STATEMENT
- 00016A 4050 D1B0      001B0 1355      STH  R5,MSGLEN
A <===== FA20 D189 4AC0 00189 02AC0 1356      AP  TASKNUM,=P'1'
- 000174 F920 D189 4AC1 00189 02AC1 1357      CP  TASKNUM,=P'2'
- 00017A 4720 2300      00300 1358      BH  ENDMSG
- 00017E F920 D189 4AC0 00189 02AC0 1359      CP  TASKNUM,=P'1'
- 000184 4780 2256      00256 1360      BE  SENDSCR4
- 000188 47F0 2300      00300 1361      B   ENDMSG
- 00018C      1363 RETURN  DS   0H
      1364 *
      1365      EXEC CICS SEND FROM(OUTMSG
      DFHECALL =X'04043000080000
      ),(FB_2,OUTLEN)
00018C 4110 D068      00068      1381 *
      1382      EXEC CICS SEND FROM(SCREEN
      DFHECALL =X'04043000080000
      2),(FB_2,MSGLEN)
0001B2 4110 D068      00068      1398 *
      1399 *      EXEC CICS RETURN
      1400 *      TRANSID(EIBTRNID)
      1401 *      COMMAREA(TSAREA)
      LENGTH(COMLEN)
    
```

Read the Source Listing Backtrace

To clearly show the program's execution path statement-by-statement, source statements that were executed from the current backtrace position are highlighted. However, when you specify a non-backtrace Source Listing option or use PF7 or PF8 to reposition the Source Listing Backtrace, the highlighting of executed statements is temporarily suspended until a backtrace PF key is entered.

When in source listing backtrace mode, the current backtrace position always appears on the line just above the first source statement. This line displays as follows:

```

===== >BACKTRACE AT #nnnnn (nnnnn -> nnnnn executed nnnnn times) <=====
    
```

The information in this line indicates the following:

- **#nnnnn**
Specifies the program statement or offset the backtrace is currently positioned at
- **nnnnn -> nnnnn**
Specifies the first and last number of the statement block
- **nnnnn times**
Specifies how many times a statement block was executed

On the Source Listing Backtrace, each relevant backtrace position is indicated in the listing by an arrow or a line:

- **====>**
Indicates the first statement in a statement block
- **<====**
Indicates the last statement in a statement block
- **====**
Indicates all other statements within a statement block

The order of the statements reflects the program's execution path. The Source Listing Backtrace displays the source of the following statements:

- The top statement is the first statement executed; the bottom statement is the most recently executed statement

Navigate Through Program Execution

You can trace your program's execution path by using the following PF keys:

- Using PF5, PF9 and PF11 to trace execution backward
- Using PF6, PF10, PF12 to trace execution forward
- Setting the Source Listing Profile *Stepping Amount*, and then using PF9 and PF10 to automatically trace the execution backward or forward

End a Source Listing Backtrace Session

To exit the Backtrace facility, press Clear or PF3. CA InterTest for CICS redisplay the Source Listing Breakpoint screen, positioned at the last breakpoint.

To toggle between the Source Listing Backtrace and the Backtrace Summary, press PF2.

Set the Code Counting Coverage Option

CA InterTest for CICS has many options for specialized testing needs. The code coverage option enables you to see the number of times an Assembler statement was executed.

The initial setting for Code Counting is OFF, which is the default. However, you can use the Source Listing Profile screen to turn Code Counting ON or OFF again at any time after you have selected monitoring for a program.

How the counts are displayed or not displayed is handled by the primary line command COUNTS and its associated parameters.

This feature causes overhead during program monitoring and should be turned off as soon as it is not longer needed.

To change the setting for Code Coverage, perform the following steps:

1. Type PROFILE on the command line and press Enter, or press PF4 to access the Profile screen from any Source Listing Screen. The Code Counting= field shows the current setting.
2. Overtyping the current value with one of the following values:
 - **YES**
Enables code counting and turns on the COUNTER display feature
 - **NO**
Removes the COUNTER display and stops the counting feature
3. Press Enter to process the new setting(s) and to return to the Source Listing Display.

4. The following screen shows the Code Coverage feature:

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING BREAKPOINT
COMMAND ==>>
Program= ASMDEMO  Option #      Stmt #      Displacement=      Margin= 01
Search=
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
      LOC  OBJECT CODE  ADDR1 ADDR2  STMT      SOURCE STATEMENT
_ 00003E 58B0 D05C      0000001 | 1212+      L      DFHEIBR,DFHEIBP
      R:B      | 1213+      USING DFHEIBLK,DFHEIBR
      | 1214+*****
      | 1215+*      END OF PROLOG CODE F
      | 1216+*****
      | 1218      PRINT NOGEN
_ 000042 4160 0232      0000001 | 1219      LA     R6, HDRLEN
_ 000046 4060 23A2      0000001 | 1220      STH   R6, OUTLEN
_ 00004A D203 D180 B008      0000001 | 1221      MVC   TSQTRAN, EIBTRNID
_ 000050 D203 D184 B010      0000001 | 1222      MVC   TSQTERM, EIBTRMID
_ 000056 956D B01A      0000001 | 1223      CLI   EIBAID, DFHCLEAR
_ 00005A 4780 2300      0000001 | 1224      BE    ENDMSG
_ 00005E 95F2 B01A      0000001 | 1225      CLI   EIBAID, DFHPF2
_ 000062 4780 399C      0000001 | 1226      BE    EXPANDEM
_ 000066 95C2 B01A      0000001 | 1227      CLI   EIBAID, DFHPF14
_ 00006A 4780 399C      0000001 | 1228      BE    EXPANDEM
_ 00006E 95F2 D188      0000001 | 1229      CLI   TSSWITCH, DFHPF2
    
```

Replacement, Protection, and Special Options

CA InterTest for CICS has many options for specialized testing needs. Just a few are explained here. For a description of all monitoring options, see [Monitoring Menu Options \(https://docops.ca.com/display/CAITSD11/Monitoring+Menu+Options\)](https://docops.ca.com/display/CAITSD11/Monitoring+Menu+Options).

You can set or remove options from the Monitoring Command Builder menus.

Replacement Options

Replacement options let you dynamically change the names of CICS resources such as programs, files, transient data queues, and temporary storage.

One reason to use replacement options is to correct errors so you can continue testing without recompiling. For example, suppose CA InterTest for CICS halts your program at an automatic breakpoint to prevent an AEIL abend caused by an incorrect file name. You can use the Replace File Name option to dynamically change the file name so testing can continue.

Another reason for using replacement options is to use different resources during testing than those hard coded in the program. For example, you can specify that the program use a test file rather than the production file named in the program.

Protection Options

When CA InterTest for CICS monitors a program, it prevents the program from modifying storage areas it does not own and issuing non-CICS instructions.

Protection options let you override the CA InterTest for CICS default rules for modifying main storage, the CSA, and load modules. The protection options, which are password protected, provide extra flexibility for special testing needs. These options allow a program to do the following modifications:

- Modify any area of main storage
- Modify areas in the CSA or CWA
- Modify a load module

These options can also prevent a program from modifying specific areas of storage not normally protected.

Special Options

Special options alter the CA InterTest for CICS standard monitoring procedures and help you design customized testing scenarios.

The No File Updating option prevents a monitored program from updating files. This option is very useful when you test a program because your file data will be unchanged at the end of each program execution. The following then happens:

- You can repeatedly test a program without restoring your files.
- Many programs can use the same file without interfering with each other's work.
- A test program can use a production file without affecting data integrity.

Other options provide additional functions. For example:

- The FOL option lets CA InterTest for CICS continue monitoring even after a program branches directly to another program (*wild branch*).
- The MXS option limits the amount of CICS storage a program can use.

- The MXR option limits the number of CICS requests a program can issue.

Composite Module Testing

Composite support lets you test all CA InterTest for CICS functions when you test *composite modules*. A composite module consists of separately compiled modules link-edited in one load module. These modules may be written in different languages and compiled at different times. Composite support lets you test and debug a called subroutine as if it were a separate program, and with full symbolic support.

For example, suppose an Assembler program has several subroutines, some written in Assembler, some written in COBOL. You can test and debug any of the subroutines just as if it were a main program by setting breakpoints, setting monitoring options, displaying the source listing, and displaying and modifying main and auxiliary storage.



Note: The following screens are for example purposes only.

Composite module monitoring is initiated by specifying the COMPOSITE command in the Source Listing Display before monitoring (PF5) is activated for the program.

```
CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==> COMPOSITE
Program= ASMPROG Option # =      Stmt #      Displacement=      Margin= 01
Search=
-----+-----
                                EXTERNAL SYMBOL DICTIONARY
SYMBOL  TYPE  ID  ADDR  LENGTH  LD ID  FLAGS
_ ASMPROG SD 0002 000000 002C32      00
_ DFHEAI0 ER 0003
_ DFHEI1  ER 0004

LOC  OBJECT CODE   ADDR1 ADDR2  STMT      SOURCE STATEMENT
                                00000  1 R0      EQU 0
                                00001  2 R1      EQU 1
                                00002  3 R2      EQU 2
                                00003  4 R3      EQU 3
                                00004  5 R4      EQU 4
                                00005  6 R5      EQU 5
                                00006  7 R6      EQU 6
```

When you specify the COMPOSITE command, CA InterTest for CICS displays the link-edit information for the main program and its subroutines. Monitor-names are specified for the programs you want to test separately. You can change this information online.

```
CA InterTest for CICS - Composite Support Builder
COMMAND ==>
Define composite support for ASMPROG and press PF5.          SCROLL: PAGE
                                                            Row 00001 of 00003

* *      *      *      *      *      *
_ Link name Monitor   Offset  Length  Language  Comments
S ASMPROG  ASMPROG    28     66C    HLASM 6.0
S ASMSUB   ASMSUB     698    448    IBMCOB 4.2
S COBOLSUB COBOLSUB   1640   16C    HLASM 6.0

PF1 Help      2          3 End      4          5 Process  6 RFind
PF7 Backward  8 Forward  9          10         11         12
```

Press PF5 to enable composite module monitoring.

Dump Analysis with CA SymDump for CICS

CA SymDump for CICS, the CA InterTest for CICS optional symbolic dump facility for z/OS users, is a powerful online tool that automatically pinpoints the source statement responsible for an abend. CA SymDump can be accessed using the ITST Primary Option Menu, Option 5 - Dump analysis.

CA SymDump for CICS was designed to complement CA InterTest for CICS, especially in production regions where programs are not usually monitored. Working together with CA InterTest for CICS, CA SymDump for CICS lets you bring a dump back to life. Familiar Source Listing screens and other facilities, such as the ability to view and modify main and auxiliary storage, make it easy to resolve application dumps.

With CA SymDump for CICS, you can perform the following tasks:

- Analyze dumps symbolically online, using CA InterTest for CICS source code listings
- Resolve CICS production dumps from any region; for example, debug production abends from your test region
- Manage your dump data set by selectively retaining the dumps you want to view and print, and discarding ones you do not need

When you use CA SymDump for CICS to view a dump, you can display the following formatted displays:

- CA InterTest for CICS Source Breakpoint at the instruction that triggered the abend
- Abend code description and probable cause
- Last 3270 screen sent to the terminal
- Program call trace (program call sequence and registers)
- CICS trace table entries, complete or filtered
- Register contents
- Lists of files and programs referenced by the transaction
- Formatted System and Data areas

Example

Suppose a program not being monitored by CA InterTest for CICS abended with an ASRA. With CA SymDump for CICS, you can display the breakpoint at the statement that triggered the abend even though CA InterTest for CICS was not active in the region when the abend occurred.

```
CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DUMP ANALYSIS
COMMAND ==>
Program= ASMDEMO  Option #      Stmt #      Displacement=      Margin= 01
                   Search=
OPTS 1 1st CSECT 2           3           4           5 Macro Cat 6 Xref
```

```

      7 Literals  8 Err msgs  9 Srch fwd 10 Srch bwd 11          12 Bkpt opts
PFKS 1 Help      2          3 End       4 Profile   5 Monitor   6 Menu
      7 Backward 8 Forward  9          10          11 Backtrace 12 Status
-----+-----
      LOC  OBJECT  CODE   ADDR1  ADDR2  STMT      SOURCE  STATEMENT
- 00016A 4050 D128      ADDR1  ADDR2  STMT      STH     R5,MSGLEN
A 00016E FA20 D101 4AC0 00101 02AC0    ==>    AP     TASKNUM,=P'1'
      ==>
      ==> CODE: ASRA
      ==>
      ==> Press PF1 for a detailed description.
      ==>
- 000174 F920 D101 4AC1 00101 02AC1    939     CP     TASKNUM,=P'2'
- 00017A 4720 2300      ADDR1  ADDR2  STMT      BH     ENDMSG
- 00017E F920 D101 4AC0 00101 02AC0    941     CP     TASKNUM,=P'1'
- 000184 4780 2256      ADDR1  ADDR2  STMT      BE     SENDSCR4
- 000188 47F0 2300      ADDR1  ADDR2  STMT      B      ENDMSG
- 00018C      945 RETURN  DS      0H
      946 *      EXEC  CICS SEND FROM(OUTMS

```

Now you can use CA InterTest for CICS to solve the problem. In this example, you might want to display the contents of TASKNUM. Or, you might want to inspect the file or look at the formatted trace table. CA InterTest for CICS provides all the tools you need to diagnose and resolve the dump.

Assembler Advanced Demo Session

CA InterTest for CICS has features designed to meet all your testing needs. The demo session in the second article illustrated the basics of working with CA InterTest for CICS. This article describes some of the advanced features.

- [Startup \(see page 61\)](#)
- [Perform the following steps before initiating the advanced demo session: \(see page 61\)](#)
- [Execute the Demo Program \(see page 67\)](#)
- [Option 01 Replace a File Control ID \(see page 68\)](#)
- [Option 02 Limit CICS Storage and Requests \(see page 72\)](#)
- [Option 03 Prevent a Program from Updating a File \(see page 77\)](#)

Only a few CA InterTest for CICS features are demonstrated in this demo session. For more information, see [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging) or refer to the Help facility.

This article consists of the following independent sections. Each section details a different advanced CA InterTest for CICS feature.

- **Option 01**
Replace a file name.
- **Option 02**
Limit the number of CICS requests issued by a program.
Limit the amount of storage obtained by a program.
- **Option 03**
Prevent a program from updating a file.
Use request breakpoints and the FILE facility.

Startup

Perform the following steps before initiating the advanced demo session:

1. Turn on the Source Listing Breakpoint facility
2. Set two unconditional breakpoints
3. Set request breakpoints at all CICS REWRITE commands
4. Execute program ASMDEMO
5. Access the Options Menu

Turn on the Source Listing Breakpoint Facility

To begin, bring up the Source Listing display for the ASMDEMO program. This was covered in the basic demo using ITST Option 1 - Source. Then select the program ASMDEMO from the Source Selection list. A quick method of displaying source for a program is given next.

1. Sign on to CICS. On a clear screen, enter the following:
list=asmdemo

2. Press Enter.
The Source Listing Display screen appears.

```
CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>>
Program= ASMDEMO  Option #          Stmt #          Displacement=      Margin= 01
                   Search=
-----+-----
Active Usings: None

  Loc  Object Code   Addr1 Addr2  Stmt          Source Statement
  ----  -
00000          1 R0      EQU    0
00001          2 R1      EQU    1
00002          3 R2      EQU    2
00003          4 R3      EQU    3
00004          5 R4      EQU    4
00005          6 R5      EQU    5
00006          7 R6      EQU    6
00007          8 R7      EQU    7
00008          9 R8      EQU    8
00009         10 R9      EQU    9
0000A         11 R10     EQU   10
0000B         12 R11     EQU   11
0000C         13 R12     EQU   12
0000D         14 R13     EQU   13
0000E         15 R14     EQU   14
0000F         16 R15     EQU   15
```

When you are debugging an Assembler program, it is sometimes easier to change the margins to view more of the source code.

1. Enter MARGIN 28 on the command line and press Enter.
or
2. Tab to the Margin field.

3. Type 28.

4. Press Enter.

CA InterTest for CICS adjusts the margin so that the display starts with column 28, as shown next.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>
Program= ASMDEMO Option # Stmt # Displacement= Margin= 28
Search=
-----+-----

```

Loc	Object Code	Addr1	Addr2	Stmt	Source Statement
1	R0	EQU	0		
2	R1	EQU	1		
3	R2	EQU	2		BASE REG
4	R3	EQU	3		BASE REG
5	R4	EQU	4		
6	R5	EQU	5		
7	R6	EQU	6		
8	R7	EQU	7		
9	R8	EQU	8		
10	R9	EQU	9		COMMAREA REG
11	R10	EQU	10		
12	R11	EQU	11		EIB REG
13	R12	EQU	12		
14	R13	EQU	13		DSA REG
15	R14	EQU	14		
16	R15	EQU	15		

Set Unconditional Breakpoints

Next, set two unconditional breakpoints. Setting unconditional breakpoints lets you halt the program so you can set options and check the contents of variables.

Begin by searching for the Assembler label NUPDTE.

1. Perform *one* of the following steps:

- Type **L NUPDTE** on the command line and press Enter.
- Tab to the Search = field, type **nupdte**, and press Enter.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>
Program= ASMDEMO Option # Stmt # Displacement= Margin=
28
Search= nupdte
-----+-----

```

Loc	Object Code	Addr1	Addr2	Stmt	Source Statement
1	R0	EQU	0		
2	R1	EQU	1		
3	R2	EQU	2		BASE REG
4	R3	EQU	3		BASE REG

5	R4	EQU	4	
6	R5	EQU	5	
7	R6	EQU	6	
8	R7	EQU	7	
9	R8	EQU	8	
10	R9	EQU	9	COMMAREA REG
11	R10	EQU	10	
12	R11	EQU	11	EIB REG
13	R12	EQU	12	
14	R13	EQU	13	DSA REG
15	R14	EQU	14	
16	R15	EQU	15	

CA InterTest for CICS displays the NUPDTE label.

2. Set an unconditional breakpoint at instruction B SENDMP00:

a. Tab to the B SENDMP00 instruction.

b. Type **U** to the left of the instruction, and press Enter. This command sets a breakpoint that stops *before* the instruction is executed. You could also type **)** to set a breakpoint that stops *after* the instruction is executed.

```
CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ===>
```

```
Program= ASMDEMO Option # Stmt # Displacement= Margi
n= 28 Search=
```

```
-----
-----
00068 REA), (FB_2,
RECLN)
```

Loc	Object	Code	Addr1	Addr2	Stmt	Source Statement
_ 001FA4	2653	NUPDTE	DS	0H		
	2654	*	EXEC	CICS	START	TRANSID
(CNTLTRAN)	2655	*	FROM	(NUPOFF)	LENGTH	(NUPOLEN) REQID
(=X'9999')	2656		DFHECALL	=X'1008F8000800005400'	(, __4, =PL4'0')	,
(CHA8,	00068			(, CHA4, CNTLTRAN)	(, _____RF, NUPOFF)	(, FB_2,
NUPOLEN)	u 001FD2	2672	B	SENDMP00		
_ 001FD6	2674	READATA	DS	0H		
	2675	*	EXEC	CICS	HANDLE	CONDITION
	2676	*	DSIDERR			

```

                2677 *                NOTOPEN (NOPEN)
                2678                DFHECALL =X'0204800008130C0000000000000000000000
00000
                00068                ,
NOPEN)
                001FF0 2689          MVC   RECKEY,
=CL10'0000000000'
                2690 *                EXEC CICS STARTBR  RIDFLD(RECKEY) DATASET
('PROTH')
                2691                DFHECALL =X'060CB000080020A000',(CHA8,
=CL8'PROTH'),,(
                00068                )),(____RF,RECKEY)
    
```

CA InterTest for CICS sets the breakpoint.

Now you are going to set a second unconditional breakpoint in ASMDemo.

1. Perform *one* of the following steps:

- Type **L LOOPRTN** on the command line and press Enter.
- Tab to the Search = field, type **looprtn**, and press Enter.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>
    
```

```

Program= ASMDemo Option #      Stmt #      Displacement=      Margin=
28
                                     Search= looprtn
    
```

```

-----
+-----
    
```

2. The LOOPRTN label appears.

3. Set an unconditional breakpoint at the LOOPRTN label:

- a. Tab to LOOPRTN.
- b. Type **U** to the left of that label.
- c. Press Enter.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>
    
```

```

Program= ASMDemo Option #      Stmt #      Displacement=      Margi
n= 28
                                     Search=
    
```

```

-----
+-----
      Loc Object Code   Addr1 Addr2  Stmt      Source Statement
_ 00213E 2818           B     SENDMP00
u 002142 2820 LOOPRTN DS     0H
_ 002142 2821           ZAP   LOOPCTR,
=P'0'
      2822 *          PRINT NOGEN
    
```

```

_ 002148 2824 CICSL00P DS 0H
          2825 * EXEC CICS ASKTIME
_ 002148 2826 DFHECALL =X'100200000800001300'
_ 00215E 2836 CP LOOPCTR,
=P'+50'
_ 002164 2837 BE MXSOPTS
_ 002168 2838 AP LOOPCTR,
=P'+1'
_ 00216E 2839 B CICSL00P
_ 002172 2841 MXSOPTS DS 0H
          2842 * EXEC CICS HANDLE CONDITION
          2843 * NOSTG
(NOSTORG)
000000 2844 DFHECALL =X'02048000082A000000000000000000000000
00068
NOSTORG)
_ 00218C 2855 ZAP LOOPCTR,=P'0'
    
```

CA InterTest for CICS sets the unconditional breakpoint.

Set Request Breakpoints

Now you are going to set request breakpoints to halt ASMDemo prior to all CICS REWRITE commands.

You can set request breakpoints to halt a program before CICS commands and macros and other calls, such as calls to DL/I or DB2. The Request Breakpoint Selection menu lets you select specific types of commands and macros at which to set request breakpoints.

You can set request breakpoints prior to all CICS commands or prior to specific commands, such as File Control or Program Control commands. Request breakpoints make it easy for you to halt your program at various points so you can check program processing. For example, you can halt the program before every READ command, single-step through the read instructions, and then check main storage to make sure the program has read the correct record.

For option 03 of the advanced demo, you need to set request breakpoints at all File Control REWRITE commands in ASMDemo. To do this, use the ITST Primary Option Menus for monitoring, as discussed next.

1. Enter the RBP command on the command line.
2. Press Enter to continue.

The Request Breakpoint Selection menu appears. Here you specify the type of request breakpoint, such as all CICS commands, File Control, Task Control, and so on.

```

CA InterTest MONITORING COMMAND BUILDER - REQUEST BREAKPOINT SELECTION 13
Set one or more types of Request Breakpoints in:
PROG=ASMDemo
_ ALL commands          _ DL/I    _ DB2    _ CALLs
_ Address, Assign,     _ Storage Control    _ BMS
  Handles, Push, Pop   _ Program Control    _ Trace Control
_ Terminal Control     _ Interval Control   _ Dump Control
    
```

CA InterTest™ and CA SymDump® - 11.0

```
s File Control          _ Task Control          _ Batch Data Interchange
_ TD Control           _ Journal Control       _ Built-In Functions
_ TS Control           _ Syncpoints            _ Sys Prog Functions
_ Web access           _ Business Trans        _ 3270 Bridge

Enter 'n' to stop only every n'th time:
Term ID (or .ANY or .NO) where breakpoints will take effect:  ----
Term ID (or .ANY) that will receive the breakpoints:         ----
User ID (or .ANY) who will execute the program:               -----

PF1 Help      2          3 End      4 Return   5          6
PF7           8          9          10         11         12
```

3. Tab to the File Control field and type an **s**.

4. Press the Enter key.

The File Control Request Breakpoint Selection menu appears. You use this menu to select the specific File Control commands where you want to set request breakpoints. In this case select REWRITE to halt the program prior to each File Control REWRITE command.

CA InterTest MONITORING COMMAND BUILDER - REQUEST BREAKPOINT SELECTION

```
Set one or more types of File Control COMMANDS IN:
PROG=ASMDemo
```

```
_ All commands
```

```
_ READ
_ WRITE
s REWRITE
_ DELETE
_ UNLOCK
_ STARTBR
_ READNEXT
_ READPREV
_ ENDBR
_ RESETBR
```

```
Enter 'n' to stop only every n'th time  ----
```

```
PF1 Help      2          3 End      4 Return   5          6
PF7           8          9          10         11         12
```

5. Tab to the REWRITE command field.

6. Type an **s** and press Enter.

CA InterTest for CICS sets request breakpoints prior to each File Control REWRITE command in the demo program and redisplay the Source Listing screen.

Check the Monitoring Status Display

Before continuing, check the Monitoring Status display for the program ASMDemo. You should see the options you set for the program, which are needed to complete the Demo Options discussed later.

1. Enter STATUS on the command line and press Enter, or press PF12 to display the Monitoring Status screen.


```
*****
****
****          CA InterTest Demo Session          ****
****          Replace File Control ID Option      ****
****
*****
```

The program attempts to read file PROTH. Because the file name has been incorrectly specified, this would cause a DSIDERR condition which would result in an AEIL abend. Instead, here is what will happen:

1. CA InterTest halts the program at an automatic breakpoint.
2. You set the Replace File Control option to change the file name without recompiling.
3. You re-execute the program from the point at which it first tried to read the file.

When this section of the demo completes, you will return to the Options Menu.

Press ENTER to continue or CLEAR to terminate

2. Press Enter.
ASMDemo halts at an automatic breakpoint.

Prevent an AEIL Abend

An example of the next screen follows. CA InterTest for CICS halted ASMDemo at an automatic breakpoint (note the A) before an AEIL abend could occur. The abend would have resulted from the EXEC CICS STARTBR command. CA InterTest for CICS highlighted the MVC instruction, the first instruction after the EXEC CICS STARTBR command.



Note: If the screen you see does not match the following example, then someone else is probably trying to perform this part of the demo session. In this case, press Enter to continue with the demo session or Clear to end the session. You can return to the Replace File Control ID option later.

```
CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= ASMDemo Option # Stmt # Displacement= Margin= 28
Search=
-----
00068 )),(_____RF,RECKEY)
A 00201E ==> MVC UNPCK,=CL4'
==>
==> CICS abend intercepted. Abend code = AEIL. EXEC CICS request was
==> STARTBR. Now stopped AFTER the CALL.
==>
==> Press PF1 for a detailed description.
==>
2707 * EXEC CICS READNEXT SET(R6) RIDFLD(RECKEY)
2708 * LENGTH(RECLEN) DATASET('PROTH')
2709 DFHECALL =X'060EF400080100B000',(CHAS,=CL8'PROTH'),(P
00068 6),(FB_2__RF,RECLEN),(_____RF,RECKEY),,(FB_2,=A(
_ 00205C 2729 MOVEREC DS
_ 00205C 2730 LH R14,RECLEN
_ 002060 2731 CVD R14,CVDAREA
_ 002064 2732 UNPK UNPCK,CVDAREA+4(4)
```

Page backward to see the data set name in the EXEC CICS STARTBR command.

1. Press PF7.
The following screen appears.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>>
Program= ASMDEMO  Option #      Stmt #      Displacement=      Margin= 28
                        Search=
-----+-----
      Loc  Object Code  Addr1 Addr2  Stmt      Source Statement
      00068      2638      DFHECALL =X'0606E0000800004000', (CHA8,CKPTNAME), (____
      _ 001FA4 2653 NUPDTE  DS      0H      REA), (FB_2,RECLLEN)
      2654 *      EXEC CICS START TRANSID(CNTLTRAN)
      2655 *      FROM(NUPOFF) LENGTH(NUPOLEN) REQID(=X'9999')
      00068      2656      DFHECALL =X'1008F8000800005400', (____4,=PL4'0'), (CHA8,
      U 001FD2 2672      B      SENDMPO0
      _ 001FD6 2674 READATA DS      0H      , (CHA4, CNTLTRAN), (____RF, NUPOFF), (FB_2, NUPOLEN)
      2675 *      EXEC CICS HANDLE CONDITION
      2676 *      DSIDERR
      2677 *      NOTOPEN (NOPEN)
      00068      2678      DFHECALL =X'0204800008130C0000000000000000000000000000000000
      _ 001FF0 2689      MVC RECKEY,=CL10'0000000000'
      2690 *      EXEC CICS STARTBR RIDFLD(RECKEY) DATASET('PROTH')
      00068      2691      DFHECALL =X'060CB000080020A000', (CHA8,=CL8'PROTH'), , (
      ) , (____RF, RECKEY)
    
```

The abend occurred because the data set name PROTH is wrong. The correct name is PROTHLF.

Change the File Name

CA InterTest for CICS lets you dynamically change an incorrect file name in the middle of your test session. This feature lets you continue testing without recompiling your program. In this case, you can correct a simple typo in program ASMDEMO without interrupting your demo session.

Follow these steps:

1. Type RO on the command line.
2. Press Enter.
The Replacement Options menu appears.
The Set Replacement Options menu lets you dynamically change a program name, file name, transient data queue name, or temporary storage ID. You can set multiple replacement options at the same time; for example, you can change both a file name and a program name. In this case, you want to change a file name.
3. Tab to the Replace file name field.
4. Specify the incorrect file name (proth) in the first column and the correct file name (prothlf) in the second column, as the following screen shows.
5. Press Enter.

```

CA InterTest MONITORING COMMAND BUILDER - REPLACEMENT OPTIONS      20
SET one or more options to replace one CICS resource name with another in:
PROG=ASMDEMO
    
```


Note: You must back up to the instruction before the EXEC CICS STARTBR command with the incorrect file name. This allows ASMDEMO to read the correct file. ASMDEMO successfully completes execution and displays the record it read.

ASMDEMO Displays the Record

Program ASMDEMO succeeded in reading a record from file PROTHLF, which is the CA InterTest for CICS Help file. ASMDEMO displays the first record in that file.

```
*****
****                                     ****
****           CA InterTest Demo Session           ****
****           Replace File Control Identifications       ****
****                                     ****
*****
```

```
Record retrieved: * 0000000000    VE0301CA InterTest HELP Fa *
                  * cility Master Menu                      *
Record length = 0960
```

You have now successfully utilized the REPLACE option

Press ENTER to continue or CLEAR to terminate

Replacing the incorrect file name in program ASMDEMO allowed you to continue testing that program without recompiling it. Of course, if one of your programs had an incorrect file name, you would eventually have to correct the file name in the original program and recompile it.

In this case, the error was caused by a simple typo. However, the Replacement Options are useful in many situations. For example, you can dynamically change a file name in order to test a program with a different file. This allows you to use a test file without altering the name of the production file coded in the program. Similarly, you can change a program name or transient data queue name to test a program or queue other than the ones that are hard coded.

You have now completed the Replace File Control ID section of the demo session. Press Enter to continue the test session, or Clear to end the session.

Option 02 Limit CICS Storage and Requests

This section of the demo session shows how to limit CICS main storage and CICS requests. Two different CA InterTest for CICS options are detailed in this part of the demo:

- The MXS option limits the amount of main storage obtained by a task. This option is useful in detecting program errors that cause a task to acquire excessive amounts of storage.
- The MXR option limits the number of CICS requests, commands and macros that a program can issue. This option is useful in detecting program loops that generate an excessively large number of CICS requests.

Before you begin this section, complete the steps outlined in Start-up (unless you did so earlier in this session). The Options Menu should be displayed.

1. Select Option 02 to request Limit CICS Storage and Requests and press Enter.
ASMDemo displays a screen that describes what will occur in this part of the demo.

```
*****
****
****          CA InterTest Demo Session          ****
****          MXR Option - Limit Number of CICS Requests      ****
****          MXS Option - Limit Amount of CICS Storage       ****
****
*****
```

The program has a loop containing a CICS request and another loop containing a GETMAIN request. Here is what will happen:

1. The program is halted at an unconditional breakpoint before the loops.
2. You set the MXS and MXR options to limit storage and CICS requests.
3. The program continues to execute.
4. CA InterTest halts the program at automatic breakpoints when the limits are exceeded.
5. You remove the options and the program completes execution.

When this section of the demo completes, you will return to the Options Menu.

Press ENTER to continue or CLEAR to terminate

2. Press Enter.
CA InterTest for CICS halts the program at the first instruction following the label LOOPRTN because you set an unconditional breakpoint there during Start-Up.
ASMDemo is halted at the first instruction after LOOPRTN. Because LOOPRTN does not have executable code, the unconditional breakpoint actually occurs before the first instruction; in this case, the highlighted ZAP instruction.
Now you can access the Special Options menu to set options you need for testing.

Set MXS and MXR Options

1. Type the SO option in the command field.
2. Press Enter.
CA InterTest for CICS displays the Special Options menu.
The Special Options menu lets you set several options to change how CA InterTest for CICS monitors a program.

You are going to set two options:

- MXS to limit the amount of CICS main storage acquired by the demo program
- MXR to limit the total number of CICS requests issued by the demo program
- Specify 50,000 bytes as the maximum amount of main storage the demo program can obtain, and specify 20 as the maximum number of CICS requests the demo program can issue.
- Tab to the MXS field and type 50000, as shown in the following screen.
- Tab to the MXR field and type 20, as shown in the following screen.

- Press Enter.

```

CA InterTest MONITORING COMMAND BUILDER - SPECIAL OPTIONS      22
SET one or more options to override the default monitoring rules in:
  PROG=ASMDemo

Enter X next to each option desired:

      Source Listing Breakpoint      (SLB)  _
      No file updating                (NUP)  _
      Reentrancy check                (RNT)  _

Follow monitoring (ON, name, NOPPT) (FOL)  _____
Number of times to be monitored          (MUS)  _____
Limit total size of CICS storage         (MXS)  50000__
Limit total number of CICS requests      (MXR)  20_____

Set local automatic breakpoint ('*', TERMID, .ANY, OFF)  _____
Limit monitoring to your TERMINAL - '*' or TERMID:      _____

Press ENTER key with data to process command or select PF key:
PF1 Help      2          3 End      4 Return    5          6
PF7           8          9          10         11         12
    
```

CA InterTest for CICS sets the MXS and MXR options and redisplay the breakpoint screen shown in the previous screen example.

- Press PF4 until you are returned to the Source Listing Breakpoint.
- Press PF5 to continue program execution. ASMDemo resumes execution.

Limit CICS Requests

The next screen that appears is the Automatic Breakpoint Display screen. CA InterTest for CICS halted ASMDemo at an automatic breakpoint because the maximum number of CICS requests (20) was exceeded.



Note: If you are working on a system with LE 370 runtimes, you could reach the automatic breakpoint for the MXS (max storage size) first. If you do, perform the demo in the following order:

1. CA InterTest for CICS Limits Acquisition of Storage removing the MXS Option
2. CA InterTest for CICS Limits CICS Requests removing the MXR Option
3. The Demo Program Executes to Completion

The instruction that triggered the automatic breakpoint is within a loop containing a CICS request.

```

CA InterTest for CICS - PROTSYM FILE  ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= ASMDemo  Option #      Stmt #      Displacement=      Margin= 28
                  Search=
-----+-----
      Loc  Object Code  Addr1 Addr2  Stmt      Source Statement
    
```


Limit Storage Acquisition

1. Press PF5 to resume program execution.
The next screen you see is an automatic breakpoint display. CA InterTest for CICS halted ASMDemo at an automatic breakpoint because the maximum main storage limit of 50,000 was exceeded.



Note: In most cases, the instruction that triggered the automatic breakpoint is within a loop containing a GETMAIN request.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= ASMDemo Option # Stmt # Displacement= Margin= 28
Search=
-----+-----
Loc Object Code Addr1 Addr2 Stmt Source Statement
 2875 *
A 0021B4 ==> CP LOOPCTR,=P'+10'
==>
==> MXS (max storage size) trigger value exceeded.
==>
==> Press PF1 for a detailed description.
==>
- 0021BA 2877 BE SENDMP00
- 0021BE 2878 AP LOOPCTR,=P'+1'
- 0021C4 2879 B GETML00P
- 0021C8 2881 NOSTORG DS 0H
- 0021C8 2882 MVC ERROR0,=CL10'NO STORAGE'
 2883 * EXEC CICS SEND
 2884 * MAP('DERROR')
 2885 * MAPSET('IN25AMP')
 2886 * ERASE
 2887 DFHECALL =X'1804D000080000000005E204000020',(CHA7,=CL
00068 ),(____RF,DERROR0),,(CHA7,=CL7'IN25AMP')
    
```

Remove the MXS Option

Remove the MXS option so that ASMDemo can continue executing. If this were your own program, you would want to examine the instructions that are causing the program to obtain excessive amounts of storage.

1. Type STATUS in the Command field and press Enter (or press PF12).
The Monitoring Status screen appears.
2. Remove the MXS option by typing R next to the MXS's breakpoint entry and pressing Enter.

```

----- CA InterTest for CICS MONITORING STATUS -----
COMMAND ==>

Type + to expand or - collapse option levels displayed below,
or R to remove option(s).

Option Description Attributes
- - ASMDemo Program monitor entry Assembler
  | Waiting at breakpoint Task 00079, ABP since 02:25 p.m.
- |-.ANY User monitoring options Active
  | Symbolic listing file PROTSYM
r | -MXS Maximum total storage 50000
.
    
```

·
·

CA InterTest for CICS removes the MXS option.

3. Press PF3 to return to the Source Listing Breakpoint screen.

Demo Program Completes Execution

Press PF5 to resume program execution.

ASMDEMO completes execution. CA InterTest for CICS returns you to the Demo Program's Options Menu.



Note: ASMDEMO continues to execute because you removed the option limiting the amount of main storage it can acquire.

Review What Happened

In this part of the demo session you set two options to help test program ASMDEMO.

- The MXR option let you limit the number of CICS requests the program could issue. When ASMDEMO exceeded this limit, CA InterTest for CICS halted the program. This feature helps you find program instructions within a loop that generate an excessive number of CICS requests.
- The MXS option let you limit the amount of main storage used by the program. When ASMDEMO exceeded the amount of storage you specified, CA InterTest for CICS halted the program. This feature helps you find commands or macros within a loop which are obtaining excessive amounts of storage.

You have now completed the Limit CICS Storage and Requests section of the demo session. Type another option number to continue with the demo session or press Clear to terminate the session.

Option 03 Prevent a Program from Updating a File

This section of the demo session shows you how to execute a program without having it update a file.

The No File Updating option is very useful when you are testing a program. Preventing a program from updating a file means your test data will be unchanged at the end of each program execution so you can test the program repeatedly without having to create test data.

Many programs can use the same test file without interfering with each other's work. You can even allow a test program to use a production file because the integrity of the file will be preserved.

Before you begin this section, complete the steps outlined earlier in this article in the section (unless you did so earlier in this session). The Options Menu appears. Also be sure that the CA InterTest for CICS checkpoint file (PROTCPF) is defined to your CICS region and that the file's current status is Open and Enabled.

1. Select Option 03 to request No File Updating and press Enter.
ASMDemo displays a screen that describes what will occur in this part of the demo session.

```
*****
****
****                      CA InterTest Demo Session                      ****
****                      No File Updating Option                          ****
****
*****
```

The program reads a record and changes data in the record. Here is what will happen:

1. CA InterTest halts the program at a request breakpoint.
2. You set the No File Updating option to prevent ASMDemo from updating the file.
3. When ASMDemo rewrites the file, the file is not updated.
4. After the rewrite you use the FILE facility to confirm that the file has not been updated.

When this section of the demo completes, you will return to the Options Menu.

Press ENTER to continue or CLEAR to terminate

2. Press Enter.
CA InterTest for CICS displays the Request Breakpoint screen.

Halt ASMDemo at a Request Breakpoint

CA InterTest for CICS halted ASMDemo prior to the first File Control REWRITE command as you specified during startup. The highlighted instruction is part of the generated EXEC CICS REWRITE command.

```
CA InterTest for CICS - PROTSYM FILE SOURCE LISTING BREAKPOINT
COMMAND ==>
Program= ASMDemo Option # Stmt # Displacement= Margin= 28
Search=
-----
00068 REA), (FB_2, RECLEM)

Loc Object Code Addr1 Addr2 Stmt Source Statement
R 001FA4 ==> NUPDTE DS 0H
2655 * EXEC CICS START TRANSID(CNTLTRAN)
2656 * FROM(NUPOFF) LENGTH(NUPOLEN) REQID(=X'9999')
2657 DFHECALL =X'1008F8000800005400', (___4, =PL4'0'), (CHA8,
00068 , (CHA4, CNTLTRAN), (____RF, NUPOFF), (FB_2, NUPOLEN)
U 001FD2 2673 B SENDMP00
_ 001FD6 2675 READATA DS 0H
2676 * EXEC CICS HANDLE CONDITION
2677 * DSIDERR
2678 * NOTOPEN (NOPEN)
2679 DFHECALL =X'020480008130C0000000000000000000000000
00068 ,NOPEN)
_ 001FF0 2690 MVC RECKEY,=CL10'0000000000'
2691 * EXEC CICS STARTBR RIDFLD(RECKEY) DATASET('PROTH')
2692 DFHECALL =X'060CB00080020A000', (CHA8, =CL8'PROTH'), (
00068 ), (____RF, RECKEY)
```

Setting request breakpoints lets you inspect the values of program variables and set options prior to all or specified CICS commands and macros and other program calls.

1. Page backward to look at the EXEC CICS REWRITE command.

2. Press PF7.
The following screen appears.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING BREAKPOINT
COMMAND ==>
Program= ASMDEMO Option # Stmt # Displacement= Margin= 28
Search=
-----+-----
Loc Object Code Addr1 Addr2 Stmt Source Statement
_ 001F14 2594 MVC FT,EIBTRMID
2595 * EXEC CICS START TRANSID(CNTLTRAN)
2596 * FROM(NUPON) LENGTH(NUPONLEN) REQID(=X'9999')
2597 DFHECALL =X'1008F8000800005400',(____4,=PL4'0'),(CHA8,
00068 , (CHA4,CNTLTRAN),(____RF,NUPON),(FB_2,NUPONLEN)
_ 001F48 2614 NUPREAD DS 0H
_ 001F48 2615 MVC RECLN,=X'2A9'
2616 * EXEC CICS READ DATASET(CKPTNAME)
2617 * INTO(NUPAREA) LENGTH(RECLN)
2618 * RIDFLD(NUPKEY) UPDATE
2619 DFHECALL =X'0602F0000800008400',(CHA8,CKPTNAME),(____
00068 REA),(FB_2__RF,RECLN),(____RF,NUPKEY)
_ 001F78 2635 NUPWRITE DS 0H
_ 001F78 2636 MVC NUPNME,=CL18'THIS IS NOT A NAME'
2637 * EXEC CICS REWRITE DATASET(CKPTNAME) FROM(NUPAREA)
2638 * LENGTH(RECLN)
2639 DFHECALL =X'0606E0000800004000',(CHA8,CKPTNAME),(____
00068 REA),(FB_2,RECLN)

```

According to the EXEC CICS REWRITE command, ASMDEMO will update the file PROTCPF (contained in the CKPTNAME field as specified in the DATASET parameter of the EXEC CICS REWRITE command).

In this case, you want to set the No File Updating option to prevent ASMDEMO from updating PROTCPF. Do this by accessing the Special Options menu.

Set the No File Updating Option

To set the No File Updating Option, do the following:

1. Type the SO option in the command field
2. Press Enter.
The Special Options menu appears.
3. Tab to the NUP field.
4. Type X, and press Enter.

```

CA InterTest MONITORING COMMAND BUILDER - SPECIAL OPTIONS      22
SET one or more options to override the default monitoring rules in:
PROG=ASMDEMO

Enter X next to each option desired:

Source Listing Breakpoint      (SLB) _
No file updating                (NUP) x
Reentrancy check               (RNT) _

Follow monitoring (ON, name, NOPPT) (FOL) -----
Number of times to be monitored (MUS) -----
Limit total size of CICS storage (MXS) -----
Limit total number of CICS requests (MXR) -----

Set local automatic breakpoint ('*', TERMID, .ANY, OFF) -----
Limit monitoring to your TERMINAL - '*' or TERMID: -----

```



```

00068          ),(____RF,RECKEY)
_ 00201E 2707      MVC  UNPCK,=CL4'
          2708 *    EXEC CICS READNEXT SET(R6) RIDFLD(RECKEY)
          2709 *          LENGTH(RECLEN) DATASET('PROTH')
          2710      DFHECALL =X'060EF400080100B000',(CHA8,=CL8'PROTH'),(P
00068          6),(FB_2__RF,RECLEN),(____RF,RECKEY),,(FB_2,=A(

```

To demonstrate that CA InterTest for CICS prevented ASMDemo from updating the file, you halted the program after the REWRITE command was executed. Now use the powerful CA InterTest for CICS FILE facility to determine whether or not ASMDemo updated the file.

1. Press PF6 Menu.
The Breakpoint Primary Option Menu appears.
2. Select option 1 - Main Menu.
CA InterTest for CICS displays the Primary Option Menu.
3. Select 4 Auxiliary storage.
The Auxiliary Storage Menu appears.

```

----- CA InterTest for CICS AUXILIARY STORAGE MENU -----
OPTION ==> 1

Select an auxiliary storage type, specifying optional criteria below.

1  Files           - Display/select files for access
2  DB2 database    - Invoke DB2 SQL interface facility
3  DL/I database   - Access DL/I database
4  TD queues       - Display/select transient data queues for access
5  TS queues       - Display/select temporary storage queues for access

Type specific or generic file/queue name(s):

protcpf_ _____
.
.

```

4. Type 1 in the Option field and protcpf for the file name.
5. Press Enter.
CA InterTest for CICS displays the File Selection menu.
6. Type an s next to the file name and press Enter.
The initial File Facility screen appears.

```

DATATYPE= FC FILEID= PROTCPF  MODE=      LOG=OFF TODEST=      PASSWORD=
FUNC=      SUBFUNC=      RETMETH=      ARGTY=      SRCHTY=
MESSAGE=
RETNRCID=      CHGLEN=
RCID=
DATA=      SIZE= 0000
FORMAT= D 00112233 44556677 8899AABB CCDDEEFF *0123456789ABCDEF*
LOC 0000 .....

```

```

-----
1 Help      2 Format C  3 End      4 BEGB      5          6 DataType DL
7 Page bwd  8 Page fwd  9 Caps Off 10 Top      11 Bottom   12

```

7. Press PF4 to begin browsing the file.
The first record in the PROTCPF file is displayed.

 **Note:** Your version of this screen might differ slightly.

```

DATATYPE= FC FILEID= PROTCPF MODE=BROWSELOG=ON TODEST= PASSWORD=
FUNC= NEXT SUBFUNC= RETMETH= ARGTP= SRCHTYP=
MESSAGE= CAIN0601 RECORD OBTAINED FOR VIEWING
RETNRCID=4040404040404040 CHGLEN=
RCID=
DATA= SIZE= 02A9
FORMAT= D 00112233 44556677 8899AABB CCDDEEFF *0123456789ABCDEF*
LOC 0000 .....
0000 40404040 40404040 4040405C 5C5C4040 *** DSORG=VSKS
0010 404040D9 C5C3D6D9 C440E3D6 40D7D9C9 RECORD TO PRI RECFM=FB
0020 D4C540E3 C8C540C3 C8C5C3D2 D7D6C9D5 ME THE CHECKPOIN LRECL=02A9
0030 E340C6C9 D3C54040 405C5C5C 5C404040 T FILE **** BLKSIZE=0000
0040 40404040 40404040 F0F3F1F4 F0F0F0F0 03140000 KEYPOS=0000
0050 00000000 00000000 00000000 00000000 ..... KEYLEN=09
0060 00000000 00000000 00000000 00000000 ..... STRN0=02
0070 00000000 00000000 00000000 00000000 .....
0080 00000000 00000000 00000000 00000000 ..... READ
0090 00000000 00000000 00000000 00000000 ..... ADD
00A0 00000000 00000000 00000000 00000000 ..... UPDATE
00B0 00000000 00000000 00000000 00000000 ..... BROWSE
00C0 00000000 00000000 00000000 00000000 ..... DELETE
-----
1 Help 2 Format C 3 End 4 ENDB 5 PREV 6 DataType DL
7 Page bwd 8 Page fwd 9 Caps Off 10 Top 11 Bottom 12
    
```

The previous screen displays the contents of the first record in both hexadecimal and character formats. It is clear from looking at the character display that the data string THIS IS NOT A NAME was not moved into this record.

- Setting the No File Updating option prevented ASMDemo from updating this file
- Before returning to ASMDemo, we will discuss the capabilities of FILE

FILE Facility

The FILE facility lets you view, update, add and delete records, and search for character strings. You can perform these functions at any time (for example, while your program is at a breakpoint). You can use FILE with VSAM and BDAM files, DL/1 and DB2 databases, temporary storage records, and transient data records. You can also use FILE when no program is executing to perform routine file maintenance.

 **Note:** Going back and forth between the Source Listing facility and the FILE facility is as easy as pressing a single key.

FILE displays records in dump format character format, vertical format, and structured format. To display the previous record in different formats, press **PF2**. For structured format, which displays records or DL/I segments on a field-by-field basis, you also must identify the program containing the structure.

Use the FILE facility when you are testing your own programs. For example, use FILE to change your test records or to create additional records in the middle of a demo session. You also can use FILE to ensure your program successfully updated a file. For more information on using FILE, see [Accessing Auxiliary Storage FILE \(https://docops.ca.com/display/CAITSD11/Accessing+Auxiliary+Storage+FILE\)](https://docops.ca.com/display/CAITSD11/Accessing+Auxiliary+Storage+FILE).

Now return to testing ASMDEMO.

Demo Program Completes Execution

1. Press Clear.
CA InterTest for CICS displays the File Selection List.
2. Press PF4 until you are returned to the Source Listing Breakpoint.
3. Press PF5 to resume program execution.
The demo program successfully completes execution, and returns you to the Demo Program's Options Menu.

Review What Happened

In this part of the demo session you took three important CA InterTest for CICS features. You were able to:

- Set request breakpoints prior to CICS commands
- Prevent a program from updating a file
- Display a record in a file

Setting request breakpoints makes it easy for you to halt your program at various points, such as before all HANDLE CONDITION commands or all Terminal Control commands. When your program halts, you can inspect main storage or auxiliary storage to detect errors and review program logic.

Preventing a program from updating a file means you can test a program repeatedly without having to create test data. Many programs can share the same test file because no program will actually change it. And, a test program can use a production file without corrupting it.

Displaying a record in a file lets you see whether your program is working correctly. The FILE facility also lets you add, delete, and update records to meet your individual testing needs and for easy file maintenance.

You have now completed the No File Updating section of the demo session. Type another option number to continue with the demo session or press Clear to terminate the session.

COBOL Primer

The main purpose of this COBOL Primer is to train new users in the basic product features used to test and debug programs. The primer also introduces some of the advanced features of CA InterTest for CICS.

We recommend that all users take the demo sessions in this article and become familiar with the features explained here. For complete information about these and other features, see [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging) . We suggest that you read the [COBOL Basic Demo Session \(see page 84\)](#) first, because it is the best way to begin learning about your product.

Getting Help

The Help facility is online documentation of product features. It makes it easy to learn and use the product. Help is available from all product screens.

To view an online summary of new features for this release, use ITST Option 8 What's New.

COBOL Basic Demo Session

This article takes you step-by-step through the basic CA InterTest for CICS demo session. Performing the demo at a terminal is the best way to begin learning about CA InterTest.

The basic demo session illustrates many of the testing and debugging tasks you will use on your own programs. For more information, see [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging).

We assume that CA InterTest for CICS has been installed on your system. If not, contact your systems programmer.

This article describes the following topics:

- [Maintain Synchronized Processing \(see page 85\)](#)
- [Demo Session Objectives \(see page 86\)](#)
- [Demo Session Scenario \(see page 86\)](#)
- [Select the Source Program from the ITST Menus \(see page 88\)](#)
- [View the Source Listing Profile \(see page 91\)](#)
- [Set Monitoring \(see page 91\)](#)
- [View Monitoring Status \(see page 92\)](#)
- [Start Program Execution \(see page 93\)](#)
- [Detect and Prevent an Abend \(see page 93\)](#)
- [Determine the Cause of the Error \(see page 94\)](#)
- [Dynamically Change the Value in TASKNUM \(see page 95\)](#)

- [COBOL Control Program Execution \(see page 97\)](#)
- [COBOL Set Unconditional Breakpoints \(see page 98\)](#)
- [What You Have Learned \(see page 100\)](#)

Maintain Synchronized Processing

Before you begin testing, you should be aware that when CA InterTest searches the symbolic files for the COBOL programs that you specify for testing, it tries to match the date and time in the symbolic file to that of the load module. If a match cannot be found, one of the following kinds of messages displays:

- A symbolic version list
- A warning message

This feature lets you maintain synchronized processing at all times by letting you select the correct symbolic version of the program that you want to test.

Symbolic Version Processing



Note: When you request a COBOL program that has no previously declared breakpoints or monitoring options set, CA InterTest matches the most recently compiled symbolic file date /time to the load module date/time.

- If an exact match is found, CA InterTest automatically selects and displays the program. However, if an exact match is found and a more recently compiled version of the program is in a PROTSYM file, CA InterTest displays the symbolic version list from which you can do the following:
 - Select the matching listing and debug the program.
 - Do not select the matching listing and cancel monitoring. Then new copy the latest version of the program into your CICS region and select the latest program version for monitoring.
- If a match is not found, CA InterTest displays the symbolic version list from which you can:
 - Select the appropriate symbolic file
 - Ignore symbolic processing for the program

The Symbolic Version List screen also explains the cause of the mismatch.

During automatic breakpoint processing, if a program has no previously selected symbolic file, CA InterTest matches the load module's date/time to a symbolic file.

- If an exact match is found, CA InterTest automatically selects and displays the program.

- If a match is not found, CA InterTest automatically selects the first symbolic file that contains the program and displays it with a warning message on the first Source Listing Breakpoint screen indicating the mismatch.

The following figure shows the Symbolic Version List screen:

```

CA InterTest for CICS - SYMBOLIC VERSION LIST

Program = COBDEMO      Load Module Date/Time = 11/12/2000 09:02:03
Loadlib = your.program.loadlib      Volume = volser

File ID      Date      Time      Language      Comments
-----
PROTSYM  11/12/2000  09.52.23  COBOL  LATEST VERSION
PROTDM0   08/31/2000  14.12.15  COBOL
.
.
.
S - Select which Symbolic file to use
-----
PFKEYS:  1 Help      2          3 No file  4          5          6
          7          8          9         10         11         12
CAIN8000 The latest Symbolic version does not match the current load module

```



Note: Once a symbolic file has been selected for a program, CA InterTest continues to use the selected file and bypasses subsequent date/time matching until all declared breakpoints and monitoring options are removed for the program, or until a CNTL=NEW, PROG=program command is executed.

Demo Session Objectives

This demo session teaches you how to do the following:

- Select a source listing for a program from the ITST menus
- Inform CA InterTest that you want to test a program (set monitoring)
- Respond to the information provided when a program error is detected
- Examine the value of a data item
- Dynamically change the value of a data item
- Halt program execution at any point (set breakpoints)
- Resume program execution

Demo Session Scenario

The CA InterTest basic demo session takes you through the following scenario:

1. You use the ITST menus to select and view the source listing of the sample program to be executed. A sample IBM Enterprise COBOL program named COBDEMO is provided for you to use.

2. You set monitoring for the sample program and view the status display of your monitoring request.
3. You exit ITST menus and initiate COBDEMO by specifying the DEMC transaction.
4. CA InterTest for CICS intercepts and prevents an ASRA abend in COBDEMO. CA InterTest halts COBDEMO at an ADD statement and displays a diagnostic message informing you that the problem was caused by improperly formatted data.
5. You examine the current value of TASKNUM, the receiving field in the ADD statement which you suspect may be the cause of the problem. You find that TASKNUM does not contain a valid packed decimal number.
6. You move a packed decimal zero into TASKNUM to correct the error.
7. You set a breakpoint to halt COBDEMO at another statement. This step is purely instructional -- it has nothing to do with correcting the ASRA.
8. You resume program execution. CA InterTest halts the program at the breakpoint you set in Step 7. You can see that when the ADD statement executed, the value of TASKNUM changed.
9. You clean up by removing the breakpoint you set. You then resume program execution and COBDEMO runs to completion.

DB2 Demo Program

DB2DEMO is a sample program that helps you to set up and verify your DB2 support for CA InterTest for CICS and CA SymDump for CICS. The demo program is written in COBOL and uses basic read-only SQL statements. The table used for queries is SYSIBM.SYSTABLES, thus, no particular database for this demo is needed. The DB2DEMO program and DEMD transaction can be used in place of the COBDEMO program and DEMC transaction as described in the Demo Session Scenario.

Follow these steps:

1. Make sure that you have successfully completed all of the steps described in Install DB2 Support section in Configuring in order to support DB2. Make sure DB2DEMO was properly installed.
2. Set up monitoring:
 - To use CA InterTest for CICS to monitor the DB2 demo program, make sure InterTest is started and DB2DEMO program is monitored by it.
 - To use CA SymDump for CICS to capture the dump of the DB2 demo program, make sure SymDump is started. If CA InterTest for CICS is also installed, make sure it does not monitor DB2DEMO.
3. Run the DB2DEMO program by invoking the DEMD transaction. The welcome screen is displayed.
4. Do one of the following:

- To abend your program without any SQL statement executed press PF2 on the welcome screen.
 - To continue and perform SQL select from SYSIBM.SYSTABLES, press Enter .
5. Following are possible errors after pressing Enter on the welcome screen:
- DB2 connection error -- Connection between the CICS region and DB2 subsystem is not established correctly.
To correct this error make sure that the CICS region is connected to the proper DB2 subsystem. Then, run DEMD transaction again.
 - SQL error -- Error occurs during the execution of a SQL statement.
To resolve this error, use the information provided on the screen (SQLCA). Find the error in the appropriate DB2 documentation and do all necessary corrections. Then run DEMD transaction again.
6. See output of SELECT SQL statement from SYSIBM.SYSTABLES table.
- Press Enter to get the next screen.
 - Press PF3 or CLEAR to end the demo.
 - Press PF2 to abend the program.
7. Abend the program:
- If DB2DEMO program is monitored by CA InterTest for CICS, the automatic breakpoint appears at "ADD 1 TO TASKNUM" statement.
 - If DB2DEMO program is not monitored by CA InterTest for CICS ASRA, the abend occurs and DB2DEMO program is terminated.
8. In case DB2DEMO is monitored for CA InterTest for CICS and you see a breakpoint appears at ADD 1 TO TASKNUM' statement, you can correct the abending statement.
- Display TASKNUM variable using KEEP or DISPAY command.
 - Edit value in TASKNUM so it is valid packed decimal number.
 - Continue program execution using GO command.
 - DB2DEMO program end screen is displayed and program ends.

Select the Source Program from the ITST Menus

Begin your session by selecting the source listing of the COBDEMO demonstration program from the ITST Primary Option Menu.

1. Sign on to CICS.
2. Type ITST on a clear screen.

3. Press Enter .

CA InterTest for CICS displays the Primary Option Menu:

```
----- CA InterTest for CICS PRIMARY OPTION MENU -----
OPTION ==>
 1 Source           - Display/select program source files/listings
 2 Monitoring       - Display/modify CA InterTest monitoring/activity
 3 Main storage     - Display/modify CICS storage areas
 4 Auxiliary storage - Display/access databases/files/queues
 5 Dump analysis    - Invoke CA SymDump CICS dump/trace capture facility
 6 Product help     - Invoke CA InterTest product help facility
 7 Status/Maintenance - Product status and maintenance functions
 8 What's new?      - Display information about CA InterTest for CICS
 X Exit            - Terminate menu processing
```

```
PF1 Help      2          3 End      4 Return    5          6
PF7           8          9          10         11         12
```

Notice the PF keys have conventional assignments for ISPF-like navigation:

PF1 Help accesses online help

- PF3 End displays the previous panel
- PF4 Return returns to the top-level menu

Now we can take a look at the source listing of the program we will be debugging.

1. Type 1 in the Option field.
2. Press Enter .
The CA InterTest Source Menu is displayed.

```
----- CA InterTest for CICS SOURCE MENU -----
OPTION ==> 1

Select a member list type, specifying optional criteria below.

 1 Source listings  - Display/select program source listings
 2 Symbolic files   - Display/select program source SYMBOLIC files

Type specific or generic program/file name(s):
(Valid mask characters are * and/or +)

c*-----
.
.
.
```

Since we know that the COBOL demo programs all start with a c, we can filter for all members that begin with that letter.

3. Type 1 in the Option field to search for program listings.

4. Tab to the first entry field for file/program names, and type **c***. The asterisk is a generic or wildcard character. It indicates that anything from this point on in a file name will satisfy the search criteria.

5. Press Enter .

The Source Listing Selection screen is displayed. It lists all program Source Listings beginning with the letter c.

```
----- CA InterTest for CICS  SOURCE LISTING SELECTION  -----
COMMAND ==>>

Type S to select a source listing.

Name      File      Created      Size Attributes      More:  +
-----
_ COBDEML  PROTSYM  08/01/2013  16:18      31 IBMCOB 3.2, no purge, composite
s COBDEMO  PROTSYM  08/01/2013  16:
03  132 IBMCOB 3.2, no purge
_ CSBIN25  PROTSYM  08/01/2013  16:12      21 IBMCOB 3.2, no purge

PF1 Help      2 Refresh      3 End          4 Return      5          6
PF7 Backward  8 Forward      9             10           11          12
```

6. Type S in the field to the left of the program name and press Enter.



Note: If the list does not contain the name of the demo program indicated previously, it may be because the user who installed CA InterTest changed the name of the CA InterTest sample programs available to you. Check with that user to find the correct names.

```
CA InterTest for CICS  - PROTSYM FILE  SOURCE LISTING DISPLAY
COMMAND ==>>
Program= COBDEMO  Option #      Stmt #      Search=      Margin= 01
-----
YEARWINDOW
(1900)
ZWB
000001 ID DIVISION.
000002 PROGRAM-ID. COBDEMO.
000003 ENVIRONMENT DIVISION.
000004 DATA DIVISION.
000005 WORKING-STORAGE SECTION.
- 000006 77 S999-FIELD1          PIC S9(3).
- 000007 77 S999-FIELD2          PIC S9(3)      VALUE +50.
- 000008 77 999-FIELD1          PIC 9(3).
- 000009 77 999-FIELD2          PIC 9(3)      VALUE 50.
- 000010 77 COMMAREA-LEN        PIC S9(4) COMP VALUE +59.
- 000011 77 LINK-COMMAREA-LEN   PIC S9(4) COMP VALUE +59.
- 000012 77 TSQ-LEN             PIC S9(4) COMP VALUE +59.
- 000013 77 REC-LEN             PIC S9(4) COMP.
- 000014 77 NUM-CHOICES         PIC S9(4) COMP VALUE +7.
- 000015 77 SUB                 PIC S9(4) COMP.
```

- The top half of the display identifies the program and provides fields for entering commands, options, statement numbers and search criteria. Depending on your session defaults, it may also display PF key assignments and available options. These have been omitted from the Source Listing displays in this article for clarity.

- The bottom half contains the source listing.



Note: The statement numbers on your screen may not match those shown in the illustrations. This will not hinder you from performing the test session.

View the Source Listing Profile

1. Type PROFILE on the Command line and press Enter (or, press PF4 Profile) to view the PF key assignments and other session options.

CA InterTest for CICS displays the Source Listing Profile for your current session. This lists the PF keys, options, and current environment settings relating to the Source Listing display.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING PROFILE
COMMAND ==>>
Program= COBDEMO  Option #          Stmt #          Search=          Margin= 01
OPTS 1 Proc div 2 Work-stor 3 Link sect 4 D-map 5 Clst/Pmap More: +
      6 Data xref 7 Proc xref 8 Err msgs 9 Srch fwd 10 Srch bwd
PFKS 1 Help 2 3 End 4 Auto prms 5 Monitor 6 Menu
      7 Backward 8 Forward 9 Next Wnd 10 11 12 Status
-----
Display window = N N (None), T (Titles), R (Registers),
                K (Keep), P (Program)
PF7/8 amount = PAGE PAGE, HALF, STOP, or a number from 1 to 9999
Step Timing = BEFORE Stop Before or After the next verb is executed
Stepping amount = 001 The number of verbs to execute
Auto-stepping = OFF ON to activate; press PF4 to change values
Source List BKPT = ON OFF to use the detailed breakpoint display
From terminal ID = U084 Terminal ID where the program will execute
BKPT terminal ID = U084 Terminal ID to receive the breakpoint displays
User ID = .ANY User ID who will execute this program
AutoKeep Display = ON OFF to deactivate
Code Counting = OFF ON to activate Code Coverage
SDF = DATA HEX for Hexadecimal/Character Format

```



Note: Verify that the AutoKeep Display feature is activated for this demo session. If not, change it from OFF to ON and press Enter.

Set Monitoring

The next step is to instruct CA InterTest to monitor COBDEMO. As part of monitoring a CICS command-level program such as COBDEMO, CA InterTest automatically prevents all CICS abends during execution. How this works is shown later on when CA InterTest prevents COBDEMO from abending because of an ASRA.

There are a number of different ways of telling CA InterTest to monitor the program. If you are currently viewing a program's source listing, you can type one of the following:

1. Type MONITOR on the Command line and press Enter.
or
2. Press PF5 (notice PF5 is listed as Monitor in the previous screen).

- Press PF5 or use the monitor command to begin monitoring the demo program. You can do this from either the Source Listing Profile or Source Listing Display. The screen momentarily flashes as the request is processed. The Source Listing Display screen then reappears.

View Monitoring Status

To verify that the demo program is being monitored:

- Type STATUS on the command line, as shown in the following screen:

```

CA InterTest for CICS  - PROTSYM FILE  SOURCE LISTING DISPLAY
COMMAND ==> status

Program= COBDEMO  Option #          Stmt #                               Margin= 01
                                Search=
OPTS 1 Proc div  2 Work-stor 3 Link sect  4 D-map    5 Clst/Pmap More:  +
      6 Data xref 7 Proc xref 8 Err msgs  9 Srch fwd 10 Srch bwd
PFKS 1 Help      2          3 End        4 Profile  5 Monitor   6 Menu
      7 Backward 8 Forward  9 Next Wnd 10          11          12 Status
-----
YEARWINDOW(1900)
ZWB
000001 ID DIVISION.
000002 PROGRAM-ID. COBDEMO.
    
```

- Press Enter

CA InterTest for CICS displays the Monitoring Status screen. This status shows the monitoring entry for the current program only.

```

----- CA InterTest for CICS  MONITORING STATUS -----
COMMAND ==>

Type + to expand or - collapse option levels displayed below,
or R to remove option(s).

Option      Description                Attributes
- - COBDEMO  Program monitor entry      IBMCOB 3.2
- - |-.ANY   User monitoring options    Active
-   |       Symbolic listing file    PROTSYM
-   | -SLB   Source listing breakpoints U023
-   |       *** End of data ***

PF1 Help      2 Refresh    3 End        4 Return     5 Collapse   6 Expand
PF7 Backward  8 Forward    9           10          11          12
    
```



Note: If the Status display does not display, you have not set monitoring correctly. Press PF3 to return to the Source Listing display, and then type MONITOR on the command line or press PF5 to set monitoring. Check the status again before continuing.

- Once you have verified that the correct demo program is being monitored, press PF3 to go back to the Source Listing. Now exit to CICS using the fastpath entry (=x), as follows:


```

- 000480 IF TASKNUM = 1
- 000481 MOVE 'DMAPASR' TO MAPNAME.
- 000482 IF TASKNUM = 2
- 000483 MOVE 'DMAPSUM' TO MAPNAME.
- 000484 IF TASKNUM GREATER 2
- 000485 GO TO SEND-END-MSG.
- 000486 GO TO REWRITE-TSQ.
- 000487 REWRITE-TSQ.
- 000488*EXEC CICS WRITEQ TS

```

Notice that CA InterTest has highlighted an ADD instruction and displayed a message below it.

Execution of that ADD instruction triggered an ASRA abend. CA InterTest prevented the abend and then displayed the diagnostic screen you are currently viewing.

When CA InterTest stops program execution, we say that it halts the program at a breakpoint. This can be done automatically, or CA InterTest halts a program at a breakpoint that was set by you, the programmer. The A to the left of the ADD statement indicates that the current halt in program execution is an Automatic breakpoint -- not one set by you.

Now look at the highlighted message. It explains that the problem was caused by improperly formatted data. Since ADD +1 TO TASKNUM triggered the breakpoint, it is likely that TASKNUM contains improperly formatted data.

So far, CA InterTest has prevented COBDEMO from abending and identified the problem.

Next, perform the following steps:

1. Determine the cause of the error.
2. Dynamically correct the error by changing the value of TASKNUM.

Determine the Cause of the Error

Next confirm that the value stored in TASKNUM is not in valid arithmetic format by examining its current value; that is, its value prior to the execution of the ADD statement that triggered the ASRA.

CA InterTest for CICS displays the current value of TASKNUM, as shown in the following screen:

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>>
Program= COBDEMO Option # Stmt # Search= Margin= 01
----- TASKNUM | ?00000.
-----+-----
- 000477 CONTINUE-TASK.
- 000478**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
A ==> ADD +1 TO TASKNUM.
==>
==> ASRA ABEND (0C7) detected and prevented. Caused by invalid decimal
==> arithmetic data format.
==>
==> Press PF1 for a detailed description.
==>
- 000480 IF TASKNUM = 1
- 000481 MOVE 'DMAPASR' TO MAPNAME.
- 000482 IF TASKNUM = 2
- 000483 MOVE 'DMAPSUM' TO MAPNAME.
- 000484 IF TASKNUM GREATER 2
- 000485 GO TO SEND-END-MSG.
- 000486 GO TO REWRITE-TSQ.

```

```
- 000487 REWRITE-TSQ.
- 000488*EXEC CICS WRITEQ TS
```

When the AutoKeep Display feature is active, CA InterTest automatically displays the value(s) of the variable(s) associated with a line of code. In this case, TASKNUM has been displayed in this specially formatted Keep window. On the left is the name of the data item; on the right is the fields' datatype appropriate display format. The question mark in the beginning of the field indicates that it contains incorrect values for its datatype.

Up to six data items will be displayed at a time within the scrollable Keep window. This feature lets you see how the values of data items change as your program executes. The Keep window remains until you remove all data items from it.



Note: A full discussion of the Keep window is described in [COBOL Debugging Programs \(see page 101\)](#). Now take a look at the contents of TASKNUM. It does not contain a valid packed decimal value. Instead, it contains low-values (binary zeros). COBOL does not allow you to add a value to a field without initializing it.

Dynamically Change the Value in TASKNUM

Now that we have identified and confirmed the cause of the problem, we can fix it.

1. Type M (modify) to the left of TASKNUM in the Keep window, as shown in the following screen.
2. Press Enter.

```
CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= COBDEMO Option # Stmt # Search= Margin= 01
-----
----- TASKNUM | ?00000.
-----+-----
- 000477 CONTINUE-TASK.
- 000478**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
A ==> ADD +1 TO TASKNUM.
```

CA InterTest for CICS generates a fill-in-the-blanks MOVE statement, as shown in the following screen.

3. Type zeros in the MOVE field, as shown next, and press Enter.



Note: You do not have to know the type of data (binary, packed, and so on) or the length of TASKNUM. The CA InterTest COBOL-like MOVE statement automatically takes care of that for you.

```
CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= COBDEMO Option # Stmt # Search= Margin= 01
```

```

                                Search=
-----
TASKNUM                          | 700000.
-----+-----

```

MOVE zeros_____

to

TASKNUM

Overtyping Underscores with a Data-Name, Figurative Constant, Alphanumeric Literal, or Numeric Literal

CA InterTest for CICS executes the MOVE statement. As a result, TASKNUM now contains a packed decimal zero. CA InterTest also displays a main storage display that shows the new value of TASKNUM, as shown in the following screen.

```

CA InterTest for CICS - MAIN STORAGE UTILITY - Termid = U002

Starting at Address =2080A138      Structure Display Format
02 TASKNUM                        | 00000. |
02 TASKNUM-CHAR                    | ...   |
02 TASK-TEXT                       |       |
03 TASK-ID-NO                      | 000.  |
03 FILLER                          |       |
03 TASK-MESG                       | THIS IS A MESSAGE |
03 FILLER                          |       |
03 TASK-DATE                       |       |
04 TASK-MM                         | 12    |
04 TASK-SL1                        | /     |
04 TASK-DD                         | 25    |
04 TASK-SL2                        | /     |
04 TASK-YY                         | 99    |
-----
PF1 Help      2          3 End      4 Return  5          6 Dump
PF7 Backward  8 Forward  9 Caps Off 10         11 Redisplay 12 Structure
CORE=MOVE ZEROS TO TASKNUM
CAIN0201 RECEIVING FIELD has been changed as shown

```

You can change the contents of a field simply by overtyping the desired bytes in the Keep window or on the main storage display. Overtyping is, of course, faster than using a CA InterTest-generated MOVE statement. However, with the MOVE statement you do not have to know the internal representation of the data. CA InterTest automatically takes care of the details for you.

Now we will return to the program listing so we can continue to test COBDEMO.

4. Press Clear.
CA InterTest for CICS redisplay the breakpoint screen without the explanation of the abend. Note the new value of TASKNUM in the Keep window.



Note: You also can correct this bug using indirect commands, which are described [Advanced Monitoring Features \(see page 123\)](#).

COBOL Control Program Execution

Now that the value in TASKNUM has been properly initialized, the next step might be to continue testing by resuming program execution. However, this is a good opportunity to learn about another important CA InterTest feature - the 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 initiate the task, and the program either runs to completion or abends.

With CA InterTest, you can control program execution in a number of ways. For example, you can set stops, called breakpoints, anywhere in your program. Four types of breakpoints you can set are unconditional, conditional, variable-change, and request breakpoints.

- When you set an *unconditional* breakpoint at a statement, the program stops just before or just after the statement is executed. (Depending on the type of unconditional breakpoint that you set.)
- When you set a *conditional* breakpoint at a statement, the program stops only if a condition you specified is met -- such as a counter equaling or exceeding some value. You can also set a conditional breakpoint to stop at any instruction when the condition you specified is met.
- When you set a *variable-change* breakpoint, the program stops at any instruction if the value of the variable you specify has changed. This is a special type of conditional breakpoint.
- When you set a *request breakpoint*, the program stops before all CICS commands, macros and other program calls, such as calls to DL/I or DB2, or just before specific CICS commands, such as all READ or WRITE commands.

What You Can Do When Your Program Is Stopped

Once a program is stopped, you can use the CA InterTest testing and debugging facilities to do the following tasks:

- Examine the source listing
- Examine and modify main and auxiliary storage to detect and correct errors
- Set and remove breakpoints
- Examine a program's backtrace, or execution path
- Keep data items in a Keep window to observe changes in their values
- Abend your task with or without a dump
- Go around a problem by resuming program 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 one verb and then stops

Now we are going to demonstrate how easy it is to control program execution by setting an unconditional breakpoint.

COBOL Set Unconditional Breakpoints

You can set unconditional breakpoints directly on the Source Listing screen just as easily as you displayed and modified TASKNUM. Enter U or) to the left of each statement where you want the application to halt program execution. The U sets a breakpoint that stops *before* the instruction is executed. The) sets a breakpoint that stops *after* the instruction is executed.

Here is how to set a "before" breakpoint at the following IF statement:

IF TASKNUM GREATER 2

1. Type **U** to the left of the statement number on the IF TASKNUM GREATER 2 line, as shown in the following screen.
2. Press Enter.

```

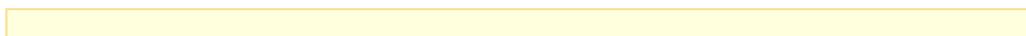
CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= COBDEMO Option # Stmt # Search= Margin= 01
-----
----- TASKNUM | 00000.
-----+-----
- 000477 CONTINUE-TASK.
- 000478*** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
A ==> ADD +1 TO TASKNUM.
- 000480 IF TASKNUM = 1
- 000481 MOVE 'DMAPASR' TO MAPNAME.
- 000482 IF TASKNUM = 2
- 000483 MOVE 'DMAPSUM' TO MAPNAME.
u 000484 IF TASKNUM GREATER 2
    
```

3. Type the **GO** command on the command line or press PF5 to continue execution from where the program is currently stopped (at the statement ADD +1 TO TASKNUM). CA InterTest for CICS resumes program execution and continues until it reaches the breakpoint you just set. CA InterTest halts the program before that statement is executed and displays the screen shown next.

```

CA InterTest for CICS - PROTSYM FILE UNCOND BEFORE BREAKPOINT
COMMAND ==>
Program= COBDEMO Option # Stmt # Search= Margin= 01
-----
----- TASKNUM | +00001.
-----+-----
- 000482 IF TASKNUM = 2
- 000483 MOVE 'DMAPSUM' TO MAPNAME.
U ==> IF TASKNUM GREATER 2
- 000485 GO TO SEND-END-MSG.
.
.
.
    
```

Notice the U to the left of the highlighted statement. It identifies the breakpoint as an unconditional "before" breakpoint. For an "after" breakpoint, the application displays a). For an automatic, conditional, variable-change, or request breakpoint, the application displays an A, C, V, or R, respectively.





Note: The value of TASKNUM is now 1 because the ADD +1 TO TASKNUM statement executed successfully.

When you are stopped at a breakpoint, you can do the following tasks:

- Scroll or search through your source listing
- Examine and modify main and auxiliary storage
- Add a data item to the Keep window using the K line command
- Set and remove breakpoints
- Examine the program's backtrace summary
- Abend your task (with or without a dump)
- Go around a problem by resuming program execution from another location. (Type G next to the instruction where you want to resume.)

When debugging your own programs, you will typically perform one or more of these activities, which are described in detail in the next article. However, COBDEMO does not have any more errors, so we are going to complete the demo session.

As you continue testing and debugging, you should clean up by removing any breakpoints no longer needed, so that when retesting the program it will not stop unnecessarily. Next, remove the unconditional breakpoint you just set so COBDEMO will not stop at this statement when it is re-executed.

Remove Breakpoints from the Status Display

If you want, you can also remove a breakpoint from the Status display:

1. Type STATUS on the command line.
2. Press Enter.

CA InterTest for CICS displays the Monitoring Status screen:

```
----- CA InterTest for CICS  MONITORING STATUS -----
COMMAND ==>

      Type + to expand or - collapse option levels displayed below,
      or R to remove option(s).

      Option      Description              Attributes
- - COBDEMO      Program monitor entry      IBMCOB 3.2
      |           Waiting at breakpoint      Task 00040, UBP since 06:48 p.m.
- - |-.ANY       User monitoring options    Active
      |           Symbolic listing file      PROTSYM
- - | -UBP       Unconditional breakpoint  #484
      |           Option ID              80258747
      |           From, to terminals    U002, U002
- - | -SLB       Source listing breakpoints U002
      |           *** End of data ***
```

PF1 Help	2 Refresh	3 End	4 Return	5 Collapse	6 Expand
PF7 Backward	8 Forward	9	10	11	12

3. Type R to the left of the unconditional breakpoint (UBP).
4. Press Enter.
An asterisk displays next to the breakpoint, indicating that a command has been processed for it.
5. Press PF2 to refresh the screen.
The unconditional breakpoint is no longer displayed.
6. Press PF3 to return to the Source Listing Breakpoint screen.

Resume Program Execution

1. Press PF5 to continue program execution.
CA InterTest for CICS resumes program execution. COBDEMO displays a screen confirming that you have successfully corrected the ASRA.
2. Press Enter.
COBDEMO displays the following screen.

```
*****
****
****          CA InterTest Demo Session          ****
****
****
*****
```

You have completed the sample CA InterTest test session. As part of this session, you:

- * displayed program source code and compiler output online
- * displayed and modified main storage
- * controlled program execution

This is just a fraction of CA InterTest's capabilities. You can also:

- * display or modify any CICS file, or DL/1, DB2, or SQL/DS database
- * set and remove many kinds of monitoring options

Press ENTER or CLEAR to complete the termination.

This screen reminds you that we have touched on just a few of CA InterTest's powerful yet easy-to-use testing and debugging facilities.

3. To complete this part of the sample test session, press Clear or Enter to clear your screen.

What You Have Learned

This demo session has taught you the basics of using CA InterTest to test a program.

You learned how to perform the following tasks:

- **Select a source program for display**
Go to ITST Menu Option 1.1. Select a program from the list.

- **Set monitoring for a program**
Use the MONITOR command or PF5 from Source Listing.
- **View a Monitoring Status display**
Use the STATUS command or PF12 from Source Listing.
- **Interpret the information the application provides when it detects an error**
View the Automatic Breakpoint and press PF1 for help.
- **Examine the value of a data item in a Keep window**
View the data in the AutoKeep Display window.
- **Modify main storage**
Overtyping the value in the Keep Window or typing D to the left of a variable. Place the cursor under the variable and press Enter for a CORE Main Storage display. On the CORE screen, overtype the values and press Enter.
- **Set an unconditional "before" breakpoint**
Type U to the left of an instruction and press Enter.
- **Remove a breakpoint from the Monitoring Status display**
Use PF12 to view Monitoring Status. Type R next to a UBP and press Enter.
- **Remove a breakpoint from the Source Listing Display**
On Source Listing, overtype U with X and press Enter.
- **Resume program execution**
Use the RESUME command or PF5.

COBOL Debugging Programs

This article provides an overview of key CA InterTest for CICS functions. For complete information, see [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging) and the Help facility.

- [Checklist of Basic Debugging Tasks \(see page 102\)](#)
- [Display Your Source Listing \(see page 102\)](#)
- [Start a Test Session \(see page 103\)](#)
- [Set and Remove Breakpoints \(see page 105\)](#)
- [Inspect and Modify Main Storage \(see page 109\)](#)
- [Keeping Data Items in the Keep Window \(see page 112\)](#)
- [Use Variable-Change Breakpoints to Detect Changing Values \(see page 113\)](#)
- [Inspect and Modify Auxiliary Storage \(see page 116\)](#)
- [Resume Program Execution \(see page 119\)](#)
- [Abend a Task \(see page 120\)](#)
- [Get Help \(see page 121\)](#)
- [End a Test Session \(see page 122\)](#)
- [Correct the Source Code \(see page 122\)](#)

Checklist of Basic Debugging Tasks

The following list highlights basic debugging tasks:

- Display the source listing.
- Set monitoring for the program.
- Set breakpoints.
- Set other monitoring options as needed.
- Initiate the program.
- End the test session.

When the program is stopped at a breakpoint, you can perform the following procedures:

- Inspect and modify main storage
- Inspect and modify auxiliary storage
- Set and remove breakpoints
- Keep data items in the Keep window to observe changes in their values
- Resume execution
- Abend the task



Note: Before you read this article, you should have performed the Basic Demo Session described in the previous article. The demo session provides a basic introduction to CA InterTest and illustrates many of the functions discussed in this article.

Display Your Source Listing

You can display your source listing online at any time from CICS through the ITST transaction or the LIST=*progname* transaction (where *progname* is the program name). The Basic Demo explains how to display a source listing using the ITST transaction. For additional information on how to use LIST=, see [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging).



Note: The standard CA InterTest transaction IDs, such as LIST, ITST, CORE, and FILE, might have been changed at your site. If any of the transactions mentioned in this article do not function as described, contact the person who installed CA InterTest.





Note: Depending on your defaults, the options and PF keys available to you might not be displayed at the top of the Source Listing Display. If this information is not displayed, you can view it by pressing PF4 to access the Profile screen, and then setting Display window to T.

Display Sections of Your Program

The options (OPTS) available at the Source Listing screen make it easy to display any section of your COBOL program.

The following table shows what program sections are displayed when you use the LOCATE command or option numbers:

Command	Option #	Section
L.PD	1	Procedure Division
L.WS	2	Working Storage Section
L.LS	3	Linkage Section
L.DM	4	Data Division Map
L.PM	5	Clist/Pmap
L.DX	6	Data Name Cross Reference
L.PX	7	Procedure Cross Reference
L.EM	8	Error Messages
L.LC	13	Local-Storage Section

Search for Strings and Labels

You can also search for and display a character string by typing a FIND string NEXT/PREV command on the command line or option 9 (Search forward) or 10 (Search backward) in the Option # field and typing the character string in the Search field. For example, the following screen illustrates how you would search backward for the string *custname*.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==> f custname prev
Program= COBDEMO Option #      Stmt #      Margin= 01
                        Search=
-----+-----
_ 000911 VSAM-REWRITE.
_ 000912   MOVE 'THIS IS NOT A NAME ' TO VSAM-NAME.
_ 000913*EXEC CICS REWRITE
000914*   DATASET(TASK-PROTCPF)
000915*   FROM(VSAM-AREA)
000916*   LENGTH(REC-LEN)
000917*   END-EXEC.
_ 000918   Call 'DFHEI1' using by content x'0606e0000700004000f0f0f5f2f4

```

Start a Test Session

To start a test session, follow these steps:

1. Turn on monitoring.

2. Execute the program.

Turn on Monitoring

To test a program, you must instruct CA InterTest for CICS to monitor it. When CA InterTest for CICS monitors a program, it detects and prevents errors before they occur, including the following violations:

- All storage violations
- All CICS abends
- Any statement that would cause a program check or other abend
- All illegal or invalid instructions that would cause CICS to crash
- All wild branches
- All violations of CICS standards

The easiest way to turn on monitoring is from the program's source listing. Perform the following steps:

1. Display the source listing as described previously. CA InterTest displays the Source Listing Display screen for the program.
2. Press PF5 to set monitoring, which remains in effect for the program until specifically removed.

Execute a Program

Once you have turned on monitoring for a program, you are ready to execute it. Perform the following actions:

1. Press PF3 to exit the Source Listing facility. If you entered Source Listing from an ITST menu, you return to that menu.
2. To exit the ITST menus, type =X in the top field and press Enter. The ITST transaction ends and you return to CICS.



Note: Before returning to CICS, you may want to set breakpoints directly on the source listing, as described in Setting and Removing Breakpoints.

3. On a clear CICS screen, type the transaction identifier of the program you are monitoring.

Once you enter the program's transaction ID, one of the following occurs:

- Your program runs to completion. The results might not be correct.

- CA InterTest for CICS stops your program at a breakpoint -- either one set by you or one automatically triggered by CA InterTest for CICS because it detected an error.

Set and Remove Breakpoints

When you test a program, it is important to be able to halt program execution at specified locations. A halt in program execution is called a *breakpoint*.

What You Can Do at a Breakpoint

When your program is stopped at a breakpoint, you can do the following actions:

- Examine the source listing
- Inspect and modify main storage
- Inspect and modify auxiliary storage
- Set and remove breakpoints
- Examine the backtrace
- Keep data items in the Keep window to observe changes in their values
- Set and remove monitoring options
- Specify indirect commands
- Resume execution
- Abend the task

We have already discussed how you can examine your source listing. The other breakpoint activities are discussed in this and the next article.

Types of Breakpoints

There are six types of breakpoints, as shown in the following table:

- **Automatic**
The program stops because CA InterTest detected and prevented an error.
- **Unconditional**
The program stops at the location you specify.
- **Conditional**
The program stops at the location you specify if a condition is met. Optionally, conditional breakpoints can be set to stop at any instruction if a condition is met.
- **Variable-change**
The program stops at any location if the value of a specified variable has changed.

- **Request**
The program stops at every CICS command or macro, or at certain CICS commands or macros, or at calls to DL/I, DB2, or software.
- **Single-step**
The program stops after executing one or more verbs.

In this section we are going to explain how to set and remove *unconditional* breakpoints as well as *variable-change* breakpoints, because you will use these the most. An automatic breakpoint occurs when CA InterTest for CICS detects an error. When a program is stopped at an automatic breakpoint, you can either correct the error or go around it. You can press PF1 to find out what caused the error and how to use CA InterTest to fix it. You set all other breakpoints.

For more information on conditional and request breakpoints and on single-stepping, see [COBOL Advanced Monitoring Features \(see page 123\)](#). The following sections explain how to set breakpoints:

- Set breakpoints at statements on the source listing
- Set breakpoints at all references to a data name
- Set breakpoints at all or selected paragraph names

Set Unconditional Breakpoints at Statements on the Source Listing

To set an unconditional breakpoint on the program source listing, follow these steps:

1. Type **U** or **)** next to the statement where you want the breakpoint. The **U** sets a breakpoint that stops *before* the statement is executed. The **)** sets a breakpoint that stops *after* the statement is executed.
2. Press Enter.
The following screen shows how to set an unconditional "before" breakpoint.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>>
Program= COBDEMO  Option #          Stmt #          Search=          Margin= 01
OPTS 1 Proc div  2 Work-stor  3 Link sect  4 D-map      5 Clst/Pmap More:  +
      6 Data xref 7 Proc xref  8 Err msgs  9 Srch fwd  10 Srch bwd
PFKS 1 Help      2          3 End       4 Profile   5 Monitor   6 Menu
      7 Backward 8 Forward  9 Next Wnd 10          11          12 Status
-----
- 000475      IF EIBAID = DFHPF14 GO TO EXPANDED-DEMO.
- 000476      GO TO SEND-FIRST-SCREEN.
- 000477      CONTINUE-TASK.
- 000478****  TASKNUM *NOTE* FIELD MUST BE INITIALIZED
- 000479      ADD +1 TO TASKNUM.
u 000480      IF TASKNUM = 1
    
```

After you set a breakpoint, CA InterTest for CICS redisplay the screen with an uppercase U or a). This character remains until you remove the breakpoint.

You can set breakpoints at any time -- before you execute the program and when the program is stopped. Although where you decide to set breakpoints depends on the specifics of your program, you might want to set breakpoints in the following places:

- At the beginning of the Procedure Division, so when the program executes you can set additional breakpoints
- At paragraph names, so you can examine the contents of variables at the start of sections
- Before a call, so you can dynamically control the program path
- At each location named in an EXEC CICS HANDLE CONDITION, so you can verify error handling

To remove an unconditional breakpoint, overwrite the U or) with an X, or type R next to the breakpoint entry on the Monitoring Status display (PF12 from Source Listing). You will want to remove breakpoints no longer needed so that when retesting the program, you are not stopped unnecessarily.

Set Unconditional Breakpoints at All References to a Data Name

You can set unconditional breakpoints at all references to a data name from the Cross Reference section.

1. Type **L.DX** in the command line or 6 in the Option # field on the Source Listing screen, as shown next, and press Enter.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==> l .dx
Program= COBDEMO Option # Stmt # Margin= 01
Search=
OPTS 1 Proc div 2 Work-stor 3 Link sect 4 D-map 5 Clst/Pmap More: +
      6 Data xref 7 Proc xref 8 Err msgs 9 Srch fwd 10 Srch bwd
PFKS 1 Help 2 3 End 4 Profile 5 Monitor 6 Menu
      7 Backward 8 Forward 9 Next Wnd 10 11 12 Status
-----
_ 000477 CONTINUE-TASK.
_ 000478**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
_ 000479 ADD +1 TO TASKNUM.
    
```

CA InterTest displays the Dataname Cross Reference section, as shown next.

2. Now type **U** or **)** to the left of the data names for which you want to set unconditional breakpoints and press Enter. The U sets a breakpoint that stops *before* the statement is executed. The) sets a breakpoint that stops *after* the statement is executed.

The following screen shows how to set an unconditional "before" breakpoint.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>
Program= COBDEMO Option # Stmt # Margin= 01
Search=
OPTS 1 Proc div 2 Work-stor 3 Link sect 4 D-map 5 Clst/Pmap More: +
      6 Data xref 7 Proc xref 8 Err msgs 9 Srch fwd 10 Srch bwd
PFKS 1 Help 2 3 End 4 Profile 5 Monitor 6 Menu
      7 Backward 8 Forward 9 Next Wnd 10 11 12 Status
-----+-----
_ DEF DATA NAMES REFERENCES
_ 32 TASK-PROTCPF M421 909 919
_ 33 TASK-PROTHLF M422 761
_ 47 TASK-SL1
_ 49 TASK-SL2
u 30 TASK-STRUCTURE 391 448 496 520 658 867 879 925 993
    
```

```

-      51 TASK-STRUCTURE-2          659
u      34 TASK-SWITCH              398 M438 M556
    
```

- To *remove* the breakpoints, overtype the U with **X**. You can remove all the breakpoints for a data name from the Cross Reference section, or you can selectively remove breakpoints at specific statements. You can also remove individual or multiple breakpoints from the Monitoring Status display. You might want to do this when you have a number of breakpoints set throughout a large program and do not want to hunt through the source listing for them.

If your source listing does not include the Cross Reference section, you can also set a breakpoint at all references to a data name from the Working-Storage section where the data name is defined.

- To display the definition, type the data name in the Search= field and press Enter.
- When CA InterTest displays the definition, type **U** to the left of the statement, as shown next, and press Enter.

```

      CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>
Program= COBDEMO Option #      Stmt #      Margin= 01
                        Search=
OPTS 1 Proc div  2 Work-stor 3 Link sect  4 D-map    5 Clst/Pmap More:  +
      6 Data xref 7 Proc xref 8 Err msgs  9 Srch fwd 10 Srch bwd
PFKS 1 Help      2          3 End       4 Profile  5 Monitor   6 Menu
      7 Backward 8 Forward  9 Next Wnd 10          11          12 Status
-----
- 000035      03 TASK-SWITCH2      PIC 99.
- 000036      03 TASK-SWITCH3      PIC X.
u 000037      03 TASKNUM          PIC S9(5) COMP-3.
- 000038      03 TASKNUM-CHAR REDEFINES TASKNUM PIC X(3).
- 000039      03 TASK-TEXT.
- 000040      05 TASK-ID-NO      PIC 9(3) COMP-3 VALUE 0.
    
```

In this example, CA InterTest sets "before" breakpoints at all references to TASKNUM.

- To remove the breakpoints, overtype the U with **X**, or remove the breakpoints from the Monitoring Status display.

Set Unconditional Breakpoints at Paragraph Names

You can set unconditional breakpoints at all procedures and all labels.

Follow these steps:

- To set unconditional breakpoints at all Procedure Names and All Labels, display the Cross Reference Table using Option # 1 (or L .DX command), type a **U** or a **)** next to the DCL heading line, and press Enter. The U sets a breakpoint that stops *before* the specified location. The **)** sets a breakpoint that stops *after* the specified location. A message prompts you to confirm the request for setting many breakpoints by pressing PF3. It is easy to set unconditional breakpoints at all or selected paragraph names from the Procedure Names section.
- To display the Procedure Names, type **L.PX** in the command line or a 7 in the Option # field on the Source Listing screen, as shown next, and press Enter.

```

      CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>  L .PX
Program= COBDEMO Option #      Stmt #      Margin= 01
                        Search=
    
```

```

OPTS 1 Proc div 2 Work-stor 3 Link sect 4 D-map 5 Clst/Pmap More: +
      6 Data xref 7 Proc xref 8 Err msgs 9 Srch fwd 10 Srch bwd
PFKS 1 Help 2 3 End 4 Profile 5 Monitor 6 Menu
      7 Backward 8 Forward 9 Next Wnd 10 11 12 Status
-----+-----
XREF(FULL)
ZWB
000001 ID DIVISION.
000002 PROGRAM-ID. COBDEMO.
000003 ENVIRONMENT DIVISION.
000004 DATA DIVISION.
_ 000005 WORKING-STORAGE SECTION.
_ 000006 77 S999-FIELD1 PIC S9(3).
    
```

CA InterTest displays the Procedure Names section.

3. Type **U** or **)** to the left of the paragraph names for which you want to set unconditional breakpoints. You can also type **U** or **)** on the PR OCEDURE NAMES line, to set breakpoints at all paragraph names, as shown in the following screen:

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>>
Program= COBDEMO Option # Stmt # Search= Margin= 01
-----+-----
_ DEFINED PROCEDURE NAMES REFERENCES
_ 982 AFTER-REWRITE
_ 858 CICS-LOOP P856
    
```

4. To remove the breakpoi nts, overtyp e the U or) w ith **X**. You can also remove the breakpoints from the Monitoring Status display.



Note: A message warns you if your specification sets more than 20 breakpoints. You can then either confirm or cancel the breakpoints.

Inspect and Modify Main Storage

The ability to inspect main storage at a breakpoint means you can determine the values of data items without using a dump. It is easy to find logic errors because you can display the value of a data item where it is defined and at any point in the code where it is referenced. And, you can *dynamically* change its value to correct errors or test other program paths.

Remember that changes to program storage are *dynamic*; that is, the change affects only the current test session. To correct program errors, you must change the source code and recompile the program.

There are many ways to inspect and modify main storage. We are going to discuss one of the easiest methods of viewing and changing program storage here. Keeping Data Items in the Keep window explains another easy way to display and modify the values of data items.

Remember, you can inspect system storage at any time, independent of program monitoring and execution.

Display the Value of a Data Item

When you are stopped at a breakpoint, you can request the display of the value of a data item directly from the source listing.

1. To display its value where it is *defined*, type D to the left of the statement defining it and press Enter, as shown in the following screen.

```

CA InterTest for CICS - PROTSYM FILE UNCOND BEFORE BREAKPOINT
COMMAND ==>
Program= COBDEMO  Option #      Stmt #                               Margin= 01
                    Search=
-----
----- TASKNUM | ?00000.
-----+-----
- 000035 03 TASK-SWITCH2 PIC 99.
- 000036 03 TASK-SWITCH3 PIC X.
U 000037 03 TASKNUM PIC S9(5) COMP-3.
- 000038 03 TASKNUM-CHAR REDEFINES TASKNUM PIC X(3).
- 000039 03 TASK-TEXT.
- 000040 05 TASK-ID-NO PIC 9(3) COMP-3 VALUE 0.
.
.
.
    
```

2. To display the value of a data item where it is *referenced*, type D to the left of the statement referencing it, place the cursor under any character in the data item, and press Enter. CA InterTest responds by displaying the contents of the data item in structured format, as shown next.

```

CA InterTest for CICS - MAIN STORAGE UTILITY - Termid = U087

Starting at Address =2080A138      Structure Display Format
02 TASKNUM | ?00000. |
02 TASKNUM-CHAR | ... |
02 TASK-TEXT | |
03 TASK-ID-NO | 000. |
03 FILLER | |
03 TASK-MESG | THIS IS A ME |
| SSAGE |
03 FILLER | |
03 TASK-DATE | |
04 TASK-MM | 12 |
04 TASK-SL1 | / |
04 TASK-DD | 25 |
04 TASK-SL2 | / |
04 TASK-YY | 99 |
| |
-----
PF1 Help 2 3 End 4 Return 5 6 Dump
PF7 Backward 8 Forward 9 Caps Off 10 11 Redisplay 12 Structure
CORE='TASKNUM'
CAIN0452 FIELD DOES NOT CONTAIN A VALID PACKED DECIMAL (COMP-3) VALUE
    
```



Note: CA InterTest displays more than just the contents of the specified data item (TASKNUM); it also displays all the items below it in the same COBOL structure.

3. Press Clear or PF3 to return to the source listing.

Modify the Value of a Data Item

You can modify the value of a data item by overtyping the bytes in the main storage display and pressing Enter. For example you could change the contents of TASKNUM in the previous figure by overtyping the question mark with a zero, CA InterTest takes care of the datatype appropriate internal conversion automatically.

Modify the Value of a Data Item Using the MOVE Statement

Overtyping the main storage display is the easiest way to change the value of a data item. However, you want to use the CA InterTest COBOL-like MOVE statement to modify a data item when you do not know the type of data (binary, packed, and so on) or its length. The MOVE statement takes care of all the details for you. Use the MOVE statement as follows:

1. To modify a data item where it is *defined*, type M to the left of the statement defining it and press Enter. For example, to modify the value of TASKNUM, you would type M instead of D.
2. If you type M to the left of multiple lines, CA InterTest generates multiple MOVE statements.
3. To modify a data item where it is *referenced*, type M to the left of the statement referencing it, place the cursor under any character in the data item, and press Enter.

CA InterTest generates and displays a fill-in-the-blanks MOVE statement, as shown in the following screen:

```

CA InterTest for CICS - PROTSYM FILE UNCOND BEFORE BREAKPOINT
COMMAND ==>>
Program= COBDEMO  Option #      Stmt #      Search=      Margin= 01
-----
_____ TASKNUM          | ?00000.
-----+-----
MOVE -----
to
TASKNUM

Overtyping Underscores with a Data-Name, Figurative Constant,
Alphanumeric Literal, or Numeric Literal

```

4. Type one of the following items in the MOVE field:
 - A variable name, such as NEXT-TASKNUM
 - One of the following COBOL-like keywords: ZEROS, SPACES, LOW-VALUES, HIGH-VALUES, QUOTES
 - An alphanumeric literal enclosed in single quotes, such as 'YES'
 - A numeric literal, with or without a leading plus, +, or minus sign, -, such as -8.
5. Press Enter to execute the MOVE statement. CA InterTest displays a main storage display showing the new value of the data item. If you execute several MOVE statements, CA InterTest shows the main storage displays one at a time. Press Clear to see the next one.
6. From the main storage display, press Clear or PF3 to return to the source listing.

Keeping Data Items in the Keep Window

The Keep window lets you display the values of an unlimited number of data items directly on the source listing. If more than six variables are selected for the Keep window, PF19 and PF20 are available to scroll forward and backward throughout the Keep window. Keeping data items in the window lets you observe changes in their values as the program executes. You also can change values by overtyping the displayed bytes.

The Keep window consists of two parts. *Dynamic* keep elements are those that consist of the variables based on the current line of code. They are automatically displayed using the AutoKeep Display profile feature. They are shown in a highlighted fashion. *Static* keep elements are those that remain in the Keep window for the duration of the execution of the program until you remove them. The *static* keep elements are displayed before the *dynamic* elements. Use scrolling to view all the elements. You can change any *dynamic* element to a *static* element simply by following the rules mentioned next.

When the Keep window is active, the options and PF key functions are not displayed. Press PF4 if you need to refer to them. When all the data items in the Keep window are removed, the options and PF key functions are redisplayed. A command line is available to help you perform CA InterTest functions when the Keep window is active.

Add a Data Item to the Keep Window (static keep element)

You can add data items to the Keep window whenever your source listing is displayed. Identify the data items whose values you want to observe either before you execute the program, or when you are at a breakpoint.

Follow these rules to add a data item to the Keep window:

- To add a data item where it is defined, type K to the left of the statement defining it, and press Enter.
- To add a data item where it is referenced, type K to the left of the statement referencing it, place the cursor under any character in the item, and press Enter.

When you add a data item to the Keep window, CA InterTest responds by displaying the data item, as shown next.

```

CA InterTest for CICS - PROTSYM FILE  ABEND DETECTED BREAKPOINT

COMMAND ==>
Program= COBDEMO  Option #          Stmt #          Search=          Margin= 01
-----
----- EIBTRNID                    | DEMC
----- EIBTRMID                    | U087
----- TASKNUM                      | ?00000.
-----+-----
- 000477 CONTINUE-TASK.
- 000478**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
A  ==>      ADD +1 TO TASKNUM.

```

Display the Data Item Structure

To display the entire COBOL structure of a data item, type a **d** to the left of the data item in the window and press Enter, as shown next.

```

CA InterTest for CICS - PROTSYM FILE  ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= COBDEMO  Option #          Stmt #          Search=          Margin= 01
-----
      EIBTRNID          | DEMC
      EIBTRMID          | U087
d_____ TASKNUM        | ?00000.
-----
+-----
- 000477 CONTINUE-TASK.
  000478**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
A  ==>      ADD +1 TO TASKNUM.
    
```

CA InterTest responds by displaying the data item and all the items below it in the same COBOL structure.

Modify the Value of a Data Item

Modify the value of a data item in the Keep window by doing the following actions:

- Overtyping the displayed bytes, just as you would overwrite the bytes in a main storage display
- Typing M to the left of the data item to use the CA InterTest MOVE statement. Both methods of modifying a data item are discussed in the section, Inspecting and Modifying Main Storage.



Note: If you try to change password-protected storage areas not owned by your program, CA InterTest prompts you to enter the password.

Remove a Data Item from the Keep Window

To remove a data item from the Keep window, type X to the left of the data item, as shown in the following screen:

```

CA InterTest for CICS - PROTSYM FILE  ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= COBDEMO  Option #          Stmt #          Search=          Margin= 01
-----
x_____ EIBTRNID          | DEMC
x_____ EIBTRMID          | U087
_____  TASKNUM          | ?00000.
-----
+-----
- 000477 CONTINUE-TASK.
  000478**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
A  ==>      ADD +1 TO TASKNUM.
    
```

In this example two data items (EIBTRNID and EIBTRMID) will be removed from the Keep window.

When all data items have been removed, the Keep window no longer displays.

Use Variable-Change Breakpoints to Detect Changing Values

You can request that CA InterTest for CICS halt execution whenever the value of a specified data item changes. This is called a variable-change breakpoint.

Set a Variable-Change Breakpoint

A quick way to set a variable-change breakpoint is from the point in the listing where it is defined. In the following example, you will set a variable-change breakpoint on TSQ-TERMID.

Follow these steps:

1. Position the listing at the definition for TSQ-TERMID by typing L TSQ-TERMID in the Command line field, and press Enter.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==> L TSQ-TERMID
Program= COBDEMO Option # Stmt # Margin= 01
Search=
OPTS 1 Proc div 2 Work-stor 3 Link sect 4 D-map 5 Clst/Pmap More: +
      6 Data xref 7 Proc xref 8 Err msgs 9 Srch fwd 10 Srch bwd
PFKS 1 Help 2 3 End 4 Profile 5 Monitor 6 Menu
      7 Backward 8 Forward 9 Next Wnd 10 11 12 Status
-----
YEARWINDOW(1900)
ZWB
000001 ID DIVISION.
000002 PROGRAM-ID. COBDEMO.
000003 ENVIRONMENT DIVISION.
000004 DATA DIVISION.
_ 000005 WORKING-STORAGE SECTION.

```

2. The listing is then positioned at the definition for the TSQ-TERMID. Type V to the left of the statement defining the TSQ-TERMID field, position the cursor on the TSQ-TERMID field, and press Enter.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>
Program= COBDEMO Option # Stmt # Margin= 01
Search=
OPTS 1 Proc div 2 Work-stor 3 Link sect 4 D-map 5 Clst/Pmap More: +
      6 Data xref 7 Proc xref 8 Err msgs 9 Srch fwd 10 Srch bwd
PFKS 1 Help 2 3 End 4 Profile 5 Monitor 6 Menu
      7 Backward 8 Forward 9 Next Wnd 10 11 12 Status
-----
_ 000027 01 TSQ-NAME.
_ 000028 03 TSQ-TRANID PIC XXXX.
V 000029 03 TSQ-TERMID PIC XXXX.
_ 000030 01 TASK-STRUCTURE.
_ 000031 03 TASK-CNTL PIC X(4) VALUE 'CNTL'.
_ 000032 03 TASK-PROTCPF PIC X(8) VALUE 'PROTCPF'.
_ 000033 03 TASK-PROTHLF PIC X(8) VALUE 'PROTHLF'.
_ 000034 03 TASK-SWITCH PIC X.

```

3. Type the DEMC transaction. The following breakpoint displays: Type the DEMC transaction. The following breakpoint displays:

```

CA InterTest for CICS - PROTSYM FILE COND BEFORE BREAKPOINT
COMMAND ==>
Program= COBDEMO Option # Stmt # Margin= 01
Search=
-- More: + ----- PFKS 19 Backward 20 Forward -
----- DMAP04AI | .....
----- DMAPBEGI | .....
----- DMAPASRI | .....
----- DMAPSUMI | .....
----- DMAPENDI | .....
----- DMAP00I | .....
-----+-----
_ 000353 MOVE EIBTRMID TO TSQ-TERMID.

```

```

000354*** THIS CODE INITIALIZES THE MAPS BECAUSE COBOL2 DOESN'T ***
V  ==>    MOVE LOW-VALUES TO DMAP04AI
    ==>
    ==>    CAIN3630 Conditional breakpoint      requested at offset .ANY
    ==>    IF='TSQ-TERMID'.NE.'TSQ-TERMID'
    ==>
-   000356          DMAPBEGI
-   000357          DMAPASRI
-   000358          DMAPSUMI
-   000359          DMAPENDI
-   000360          DMAP00I

```

 **Note:** the listing is positioned on the statement following that which actually changed TSQ-TERMID, because the field is examined at the beginning of each new COBOL statement. The listing is positioned on the statement following that which actually changed TSQ-TERMID, because the field is examined at the beginning of each new COBOL statement.

A variable-change-breakpoint may be set any time during execution, or from a listing of your program before execution begins. You may find it useful to also set a static keep on the field which you are interested in watching for a change. In this way you will be able see how it changed automatically each time a variable change breakpoint occurs. Each variable change breakpoint is identified by the abbreviation VBP on the Status Report and Monitoring Status display. Variable-change breakpoints are actually a special type of conditional breakpoint. CA InterTest for CICS checks for the condition at any statement, as shown in the following example:

```
IF 'data-item'.NE.data-item'
```

which means:

```
IF data-item (current) NOT EQUAL TO data-item (value when breakpoint set)
```

To explore additional ways to set and use conditional breakpoints, see the later articles in this primer.

Remove a Variable-Change Breakpoint

From the variable-change breakpoint display, overtype the V with an X and press Enter to remove the breakpoint. You can also remove the breakpoint from the Monitoring Status display. Type R next to the VBP option to be removed, and press Enter. The following screen shows removing the variable-change breakpoint set on the variable TASKNUM in the program COBDEMO.

```

----- CA InterTest for CICS  MONITORING STATUS  -----
COMMAND ==>

Type + to expand or - collapse option levels displayed below,
or R to remove option(s).

Option      Description                      Attributes
- - COBDEMO Program monitor entry              COBOL
  |         Waiting at breakpoint    Task 00059, CBP since 10:54 a.m.
- - |-.ANY  User monitoring options    Active
  |         Symbolic listing file    PROTSYM
r - | -VBP  Variable change breakpoint Location=.ANY
  |         Option ID                 23DEBD7B
  |         Condition                  IF='TASKNUM'.NE.'TASKNUM'
  |         Compare type                Comp-3 (packed decimal)

```

```

_      |      From, to terminals      G001, G001
      | -SLB  Source listing breakpoints G001
      |      *** End of data ***
    
```

```

PF1 Help      2 Refresh      3 End          4 Return      5 Collapse    6 Expand
PF7 Backward  8 Forward      9             10           11           12
    
```

Inspect and Modify Auxiliary Storage

You can use CA InterTest to inspect and modify the following items:

- VSAM and BDAM files
- DL/I, DB2, and SQL/DS databases
- Temporary storage
- Transient data

You can inspect and modify auxiliary storage at any time, even if a program is executing. This capability lets you maintain files and databases without writing one-time programs. You can perform any VSAM, BDAM, or DL/I function and most SQL functions. The ability to inspect and modify auxiliary storage when a program is stopped at a breakpoint lets you change test data in the middle of a test session.

Any changes you make are permanent; that is, changes to the file remain after the test session ends.

Inspect Auxiliary Storage

To access the auxiliary storage facility, select 4 Auxiliary storage from the CA InterTest Primary Option Menu, then specify the file you want to examine on the Auxiliary Storage Menu.

```

-----CA InterTest for CICS  AUXILIARY STORAGE MENU  -----
OPTION ==> 1

Select an auxiliary storage type, specifying optional criteria below.

1  Files           - Display/select files for access
2  DB2 database    - Invoke DB2 SQL interface facility
3  DL/I database   - Access DL/I database
4  TD queues       - Display/select transient data queues for access
5  TS queues       - Display/select temporary storage queues for access

Type specific or generic file/queue name(s):
  (Valid mask characters are * and/or +)

file_____
    
```

Alternatively, if you are at a clear CICS screen, you can invoke the FILE transaction. CA InterTest displays the following:

```

DATATYPE= FC FILEID=          MODE=          LOG=ON  TODEST=          PASSWORD=
FUNC=      SUBFUNC=          RETMETH=          ARGTYP=          SRCHTYP=
MESSAGE=
RETNRCID=          CHGLEN=
RCID=
DATA=
FORMAT= D 00112233 44556677 8899AABB CCDDEEFF *0123456789ABCDEF*
LOC 0000  .....
    
```

CA InterTest for CICS V10.0

Copyright © 2016 CA. All rights reserved

```
-----
1 Help      2 Format C  3 End      4 BEGB      5           6 DataType DL
7 Page bwd  8 Page fwd  9 Caps Off 10 Top      11 Bottom   12
```



Note: If the FILE transaction does not display the previous screen, it is probably because the person who installed CA InterTest changed the name of the FILE transaction. Contact that person to find out the correct name.

After you type the file or database and, optionally, the record or segment, CA InterTest displays the data. A sample VSAM record in dump format follows.

```

DATATYPE= FC FILEID= INVNTRY  MODE=      LOG=OFF  TODEST=      PASSWORD=
FUNC=      SUBFUNC=      RETMETH=      ARGTYP=      SRCHTYP=
MESSAGE= RECORD OBTAINED FOR VIEWING
RETRNCID=F0F0F0F0F0F0F0F5          CHGELEN=
RCID=C`000000005'
DATA=                                SIZE= 0050
FORMAT= D 00112233 44556677 8899AABB CCDDEEFF *0123456789ABCDEF*
LOC 0000 .....
0000 F0F0F0F0 F0F0F0F0 F0F54040 0087338F 0000000005 DSORG=VSKS
0010 F0F3F9F9 0407835C 40404040 40404040 0399.... RECFM=VB
0020 40404040 40404040 C1C2C340 C3D6D9D7          ABC CORP LRECL=0050
0030 F0F9F1F2 F8F74040 0000598C 40404040 091287     BLKSIZE=0000
0040 D1D6C3D5 404040E2 D4C9E3C8 40404040 JOHN SMITH KEYPOS=0000
0050                                           KEYLEN=0A
0060                                           STRNO=01
0070
0080                                           READ
0090                                           ADD
00A0                                           UPDATE

```

```
-----
1 Help      2 Format C  3 End      4 BEGB      5 PREV      6 DataType DL
7 Page bwd  8 Page fwd  9 Caps Off 10 Top      11 Bottom   12
```



Note: The data appears in both hexadecimal and character format.

To return to the source listing, press Clear or PF3.

Modify Auxiliary Storage

To modify the record or segment, overwrite the hexadecimal or character data and press Enter. For example, you could overwrite JOHN SMITH with DALE COOPER on the sample display in the previous figure.

To rewrite the modified record, type PUT in the FUNC= field and press Enter.


```

000702*          DSIDERR
000703*          NOTOPEN (NOT-OPEN)
000704*          END-EXEC.
_ 000705          Call 'DFHEI1' using by content x'0204800007130c00000000000000
_ 000706-        '0000000000000000f0f0f3f8f7404040' end-call
.
.
.

```

When you press Enter, execution resumes from READ-DATASET, the location you identified with G.

Single-Stepping

Single-stepping lets your program execute one COBOL verb and then stop again. You can adjust the step amount so the program executes a specified number of verbs between stops.

To single-step, type NEXT in the command line and press Enter or press PF10 from the source listing. Your program executes the next verb and then stops.

To single-step from a different program location:

1. Display the statement where you want to resume execution. For more information, see [Display Your Source Listing](#).
2. Type G to the left of that statement.
3. Type NEXT in the command line and press Enter or press PF10.

To adjust the step amount, perform the following steps:

1. Press PF4 to access the Source Listing Profile screen.
2. Change the number in the Stepping amount field.
3. Press Enter. CA InterTest changes the step amount and redisplay the source listing. If Titles are set to on (an option on the Source Listing Profile screen), you see the new step amount displayed next to PF10 at the top of the Source Listing Breakpoint screen.

Abend a Task

When your program is stopped at a breakpoint, you can abend your task rather than resume execution. Type ABEND in the Command line of any Breakpoint display. CA InterTest displays the screen shown next.

```

----- CA InterTest for CICS ABEND BREAKPOINTED TASK -----
COMMAND ==>

Type an abend code, then press ENTER.

Abend Code  ____  Abend code options are:
                blanks      Normal abend, no dump
                XXXX        Abend exits cancelled, no dump
                your code    Your abend code, dump taken
.
.
.

```

- To cancel the abend, press PF3. CA InterTest redisplay the Primary Option Menu.
- To abend *without* a dump:
 - For a normal abend, press Enter.
 - To cancel all program exits, specify code XXXX and press Enter.
- To abend *with* a dump, specify a four-character dump code and press Enter. The dump code you specify cannot be XXXX and cannot begin with the letter A.

For more information on the status of your program after abending, see [End a Test Session \(see page 122\)](#).

Get Help

Help is always available when you are using CA InterTest. You can use Help to initiate CA InterTest. Type Help on a clear screen under CICS.

Get Help in Using CA InterTest for CICS

When you are using CA InterTest, you can press PF1 at any time for information on CA InterTest functions. For example, the following screen lists the Help topics for the Source Listing facility.

```

                CA INTERTEST - INTERACTIVE HELP FACILITY -
TUTORIAL: SOURCE LISTING FACILITY - FOR COBOL PROGRAMS

05 Set Monitoring                45 Set/Remove other monitoring Options
10 Locating areas in the listing 50 Displaying data items
20 Search for a Character string 55 Modify the value of data items
25 Specify Indirect Commands    60 Resume TASK execution
30 PF key definitions           65 Abending your TASK
32 Primary Line Commands        70 PROFILE, AUTOSTEP, AUTOKEEP Options
35 Set the Display Margins      75 Code Coverage Option
40 Set/Remove Breakpoints       80 View Program Backtrace

                                                                    (end)
-----
ENTER A SELECTION CODE FROM THE MENU, OR N FOR NEXT PAGE, P FOR PRECEDING PAGE,
M FOR RETURN TO PREVIOUS MENU, CLEAR TO EXIT, OR MESSAGE NUMBER. ==> 05
    
```

Help for Line Commands

Notice that selection code 32 lists Line Commands. These are the simple commands that you can enter on the Command line of a Source Listing Display, Breakpoint Display, or CA SymDump Breakpoint display, and include the commands abend, monitor, and status.

Get Help to Correct an Error

Whenever your program is stopped at an automatic breakpoint, CA InterTest displays a short message identifying the reason for the breakpoint. You can request help by pressing PF1 for information on what caused the error that triggered the breakpoint and how to fix it. The following screen explains how to correct an AEIL abend:

```

                CA INTERTEST - INTERACTIVE HELP FACILITY -
TUTORIAL: ERROR MESSAGE AEIL

The dataset name referred to in the DATASET option cannot be found in the FCT.
    
```

Your program did not have a Handle Condition for this error. Or a second Handle Condition without a routine for this error superseded the one that had a routine for this error.

WHAT YOU CAN DO: If the named file was entered incorrectly and the one you wanted exists, you may use the Replace File Option to dynamically replace the file name and then use the resume task facilities to execute the CICS request again. To perform the above functions from the Source Listing Breakpoint screen you would:

1. Key =20s in the Option # field and press ENTER.
2. Tab to Replace file name: Key in the incorrect file name.
3. In the next field, key in the correct file name and press ENTER.
4. Press CLEAR to return to your Source Listing Breakpoint screen.
5. Key in G to resume execution at the beginning of the EXEC CICS command and press ENTER.

(continued)

 ENTER N FOR NEXT PAGE, P FOR PRECEDING PAGE, F FOR FIRST PAGE, OR -
 M FOR RETURN TO PREVIOUS MENU, CLEAR TO EXIT, OR MESSAGE NUMBER. ==> N



Note: When you leave and then return to an automatic breakpoint screen, the error message may no longer appear. For help on the cause of the error, press Clear to redisplay the error message before pressing PF1.

End a Test Session

After your program completes execution or you abend your task, you return to CICS.

The following table shows the status after testing of a program area:

Program Area	Status after Testing
program source code	not modified
file/database updates by the program	permanent
file/database updates you made	permanent

CA InterTest monitoring, breakpoints, and other options set for the program remain in effect until they are specifically removed.

Correct the Source Code

At some point you might want to correct the source code and continue testing from a clean version of the program.

After you have modified the source code and recompiled the program, you can load the new copy of the program by using the New Program Copy option on the Program Monitoring menu (ITST 2.1). The next article explains how to set options on the Program Monitoring menu.

The New Program Copy option resets symbolic breakpoints and other monitoring options for the recompiled program. It also resets the PPT entry to the program's new library address.

COBOL Advanced Monitoring Features

This article describes just a few of the more commonly used features. For information about CA InterTest features, see [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging) and the Help facility.

- [Set Options from the Monitoring Menus \(see page 123\)](#)
- [Set and Remove Conditional Breakpoints \(see page 124\)](#)
- [Remove Conditional Breakpoints from the Source Listing \(see page 125\)](#)
- [Request Breakpoints \(see page 126\)](#)
- [Backtrace Facility \(see page 128\)](#)
- [Set the Code Counting Coverage Option \(see page 131\)](#)
- [Statement Trace Facility \(see page 132\)](#)
- [Indirect Commands \(see page 134\)](#)
- [Replacement, Protection, and Special Options \(see page 138\)](#)
- [Composite Module Testing \(see page 139\)](#)
- [Dump Analysis with CA SymDump \(see page 140\)](#)

Set Options from the Monitoring Menus

The CA InterTest monitoring menus are the easiest way to set many monitoring options. You can access these menus from the Primary Option Menu by selecting option 2 - Monitoring, as shown in the following screen:

```

----- CA InterTest for CICS PRIMARY OPTION MENU -----
OPTION ==> 2
 1 Source           - Display/select program source files/listings
 2 Monitoring       - Display/modify CA InterTest monitoring/activity
 3 Main storage     - Display/modify CICS storage areas
 4 Auxiliary storage - Display/access databases/files/queues
 5 Dump analysis    - Invoke CA SymDump CICS dump/trace capture facility
 6 Product help     - Invoke CA InterTest product help facility
 7 Status/Maintenance - Product status and maintenance functions
 8 What's new?      - Display information about CA InterTest for CICS
 X Exit            - Terminate menu processing

```

CA InterTest displays the Monitoring Menu (shown next), from which you can select a wide variety of different monitoring options. Use online help or [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging) for a detailed explanation of each of these options.

```

----- CA InterTest for CICS MONITORING MENU -----
OPTION ==>
 1 Programs         - Display/modify program monitoring options
 2 Transactions     - Display/modify transaction monitoring options
 3 Terminals        - Display/modify terminal monitoring options
 4 Status           - Display/remove current monitoring options
 5 Active tasks     - Display/purge active monitored tasks
 6 System-wide     - Display/modify global system-wide options
.
.
.
```

Option 1 Programs access the Program Monitoring Menu, from which you can set and remove any option for a program, and request a status display.

The Program Monitoring menu, shown in the following screen, is probably the most frequently used menu during a testing session:

```

----- CA InterTest for CICS PROGRAM MONITORING -----
COMMAND ==>

Type information and S to set or R to remove option(s) below.

Program . . cobdemo_      Program name (or .ALL, .OPTIONS or generic)
User ID . . _____    User (or .ANY) for whom the program is monitored

Option      Description                                          More: +
- Status    Display and/or remove monitoring options (S only)
- Monitor   Monitoring (R removes monitoring and all options previously set)
- UBP       Unconditional breakpoints (specific program only)
- CBP       Conditional breakpoints (specific program only)
- RBP       Breakpoints for CICS, DB2, DL/I or external CALL requests
- Stmt Trace Statement tracing and data monitoring (COBOL only)
- New copy  Fetch new copy of program and reset monitoring options (S only)
- Commands  Indirect commands defined for a specific COBOL or PL/1 program
- Replace   CICS resource name replacement options
- Protect   Storage protection monitoring options
- Special   Other options (storage allocation, file updating, etc.)
- Composite Monitor multi-CSECT program's separately compiled components

PF1 Help    2          3 End      4 Return   5          6
PF7 Backward 8 Forward  9         10        11        12
    
```

Quick access to this menu is available before or during a session, as described in the following table.

The following list contains the fastpath entries that you can use to access the Program Monitoring menu:

- CICS, clear screen: ITST 2.1
- Primary Option Menu, Option field: 2.1
- Source Listing display, Command field: =2.1
- Breakpoint display, Command field: =1.2.1

After processing any program options from this menu, press PF3 to exit back to your test session. If you entered directly from CICS, you will exit back to the ITST Primary menu. From a source listing or breakpoint, PF3 takes you back to that display.

Set and Remove Conditional Breakpoints

Conditional breakpoints are breakpoints that take effect when a specific condition is met, such as when a counter exceeds a value. These breakpoints can help you isolate complex problems.

To set a conditional breakpoint, type C to the left of the statement where you want to set the breakpoint, as shown next, and press Enter.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>
Program= COBDEMO Option #      Stmt #      Margin= 01
Search=
OPTS 1 Proc div 2 Work-stor 3 Link sect 4 D-map      5 Clst/Pmap More: +
      6 Data xref 7 Proc xref 8 Err msgs   9 Srch fwd  10 Srch bwd
PFKS 1 Help    2          3 End      4 Profile   5 Monitor   6 Menu
      7 Backward 8 Forward  9 Next Wnd 10         11        12 Status
-----
    
```

```

_ 000435      Go to  GEN-ERR depending on dfheigdi.
_ 000436
_ 000437 NO-OPTS.
c 000438      MOVE SPACE TO TASK-SWITCH.
_ 000439      MOVE ZERO  TO TASK-SWITCH2.
_ 000440      MOVE SPACE TO TASK-SWITCH3.

```

CA InterTest displays the menu on which you define the condition. For the following example, the breakpoint halts program execution only if TASK-SWITCH is equal to 3.

```

                CA-InterTest MONITORING COMMAND BUILDER - CONDITIONAL BREAKPOINT

Enter LEFT SIDE
  Data Name task-switch_____

Enter OPERATOR (EQ, NE, GT, LT, GE, LE):  eq

Enter RIGHT SIDE
  Data Name _____
  OR
  Literal 3 _____

_ ENTER S to Drop monitoring on a true condition

For location:
      000498      MOVE SPACE TO TASK-SWITCH.

Press PF9 to go to complex conditional screen if necessary
PF1 Help      2          3 End      4 Return  5          6
PF7           8          9 Complex 10          11         12

```

Conditional breakpoints are indicated by a C next to the line of code in your source listing.

Remove Conditional Breakpoints from the Source Listing

You can easily remove a conditional breakpoint from where it is displayed in your source listing or from the monitoring status report. To remove a conditional breakpoint from your source listing, overwrite the C with an x and press Enter, as shown in the following screen:

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>
Program= COBDEMO Option # Stmt # Margin= 01
Search=
OPTS 1 Proc div 2 Work-stor 3 Link sect 4 D-map 5 Clst/Pmap More: +
      6 Data xref 7 Proc xref 8 Err msgs 9 Srch fwd 10 Srch bwd
PFKS 1 Help 2 3 End 4 Profile 5 Monitor 6 Menu
      7 Backward 8 Forward 9 Next Wnd 10 11 12 Status
-----
_ 000435      Go to  GEN-ERR depending on dfheigdi.
_ 000436
_ 000437 NO-OPTS.
x 000438      MOVE SPACE TO TASK-SWITCH.
_ 000439      MOVE ZERO  TO TASK-SWITCH2.
_ 000440      MOVE SPACE TO TASK-SWITCH3.

```

Remove Conditional Breakpoints from the Status Report

You can always remove any breakpoint from the Monitoring Status report. To remove this conditional breakpoint from your program, perform the following steps:

1. Type STATUS on the Command line to display the Monitoring Status report for the current program.
2. Tab to the entry for the Conditional Breakpoint (CBP option) you want to remove, type R to the left of its entry, and press Enter.

```

----- CA InterTest for CICS  MONITORING STATUS  -----
COMMAND ==>

      Type + to expand or - collapse option levels displayed below,
            or R to remove option(s).

      Option      Description      Attributes
      - - COBDEMO Program monitor entry  IBMCOB 3.2
      - - |-.ANY  User monitoring options  Active
      r - |       Symbolic listing file  PROTSYM
      |       Conditional breakpoint    #438
      |       Option ID                 2E72B0DD
      |       Condition                 IF='TASK-SWITCH'.EQ.C'3'
      |       Compare type              Pic X (character strings)
      |       From, to terminals        U056, U056
      - - | -SLB  Source listing breakpoints  U056
      |       *** End of data ***

PF1 Help      2 Refresh      3 End          4 Return      5 Collapse    6 Expand
PF7 Backward  8 Forward      9             10           11           12
    
```

CA InterTest places an asterisk to the left of the breakpoint to indicate that the removal was processed.

3. Press PF3 to return to your Source.

Request Breakpoints

Request breakpoints are breakpoints that take effect prior to CICS commands and macros and other program calls, such as calls to DL/I or DB2. With one specification, you can set breakpoints at all CICS commands or at specific commands, such as all REWRITE commands.

Use request breakpoints to stop a program before the following commands:

- Every CICS command or macro
- Specific types of CICS commands, such as all File Control or Program Control commands
- Specific commands, such as all READ or WRITE commands

Set Request Breakpoints

To set request breakpoints, perform the following steps:

1. Select option - 2 Monitoring on the Primary Option Menu. CA InterTest displays the Monitoring Menu.
2. Select option - 1 Programs. CA InterTest displays the Program Monitoring screen.

```

----- CA InterTest for CICS  PROGRAM MONITORING  -----
-----
COMMAND ==>
    
```

Type information and S to set or R to remove option(s) below.

```

Program . . cobdemo_      Program name (or .ALL, .OPTIONS or generic)
User ID  . . _____   User (or .ANY) for whom the program is monitored

Option      Description                                          More:  +
- Status    Display and/or remove monitoring options (S only)
- Monitor   Monitoring (R removes monitoring and all options previously set)
- UBP       Unconditional breakpoints (specific program only)
- CBP       Conditional breakpoints (specific program only)
s RBP       Breakpoints for CICS, DB2, DL/I or external CALL requests
- Stmt Trace Statement tracing and data monitoring (COBOL only)
- New copy  Fetch new copy of program and reset monitoring options (S only)
- Commands Indirect commands defined for a specific COBOL or PL/I program
- Replace   CICS resource name replacement options
- Protect   Storage protection monitoring options
- Special   Other options (storage allocation, file updating, etc.)
- Composite Monitor multi-CSECT program's separately compiled components

PF1 Help    2                3 End          4 Return       5                6
PF7 Backward 8 Forward       9              10             11             12
    
```

3. In the Program field, type the name of the program for which you want to set request breakpoints.
4. Type S to the left of the RBP option (request breakpoints). Press Enter. CA InterTest displays the Request Breakpoint Selection menu on which you specify the type of request breakpoint, such as: all CICS commands, File Control and Task Control.

CA InterTest MONITORING COMMAND BUILDER - REQUEST BREAKPOINT SELECTION 13

```

                Set one or more types of Request Breakpoints in:
                PROG=COBDEMO

- ALL commands          - DL/I    - DB2    - CALLs
- Address, Assign,     - Storage Control  - BMS
  Handles, Push, Pop   - Program Control  - Trace Control
- Terminal Control     - Interval Control - Dump Control
- File Control         - Task Control     - Batch Data Interchange
- TD Control          - Journal Control  - Built-In Functions
- TS Control          - Syncpoints       - Sys Prog Functions
- Web access          - Business Trans   - 3270 Bridge

Enter 'n' to stop only every n'th time:
Term ID (or .ANY or .NO) where breakpoints will take effect:  -----
Term ID (or .ANY) that will receive the breakpoints:          -----
Statement no. of indirect command(s) to be executed:          -----
User ID (or .ANY) who will execute the program:                -----

PF1 Help    2                3 End          4 Return       5                6
PF7         8                9              10             11             12
    
```

Subsequent menus prompt you for the specific command, such as all File Control commands, READ, WRITE, REWRITE, and so on.



Note: Type X in front of each type of command for which you want to set request breakpoints.

After setting your request breakpoints, PF4 returns you to the Primary Option Menu. If you came from a Source Listing, PF3 returns you to your source.

Remove Request Breakpoints

To remove request breakpoints, perform the following steps:

1. From the Source Listing Display screen or breakpoint display, type STATUS on the Command line and press Enter. CA InterTest displays the Monitoring Status display showing all options for the current program.
2. Type an **r** to the left of the request breakpoint (RBP) you want to remove and press Enter. CA InterTest places an asterisk to the left of the breakpoint to indicate that the command has been processed.
3. Press PF3 to return to the Source Listing screen or Breakpoint display.

Backtrace Facility

When your program stops at a breakpoint, you might find it useful to examine the Backtrace Summary and the Source Listing Backtrace. The backtrace facility shows the logic flow of the program and explains how processing reached this point.

The backtrace facility lets you do the following actions:

- View the backtrace summary screen for a (high-level) summary of the program's execution
- Use special PF keys to step through the program's execution path
- Reposition the source listing to view the program's execution path from a different backtrace position

In addition to the Backtrace PF keys, all other source listing options, such as the Keep window, setting and removing breakpoints, and the ability to scroll backward and forward from the current source listing position are available from the Source Listing Backtrace screen.

Access the Backtrace Summary

To view the Backtrace Summary, press PF11 from the Source Listing Breakpoint screen, or, type BTRACE in the Command line and press Enter. CA InterTest positions the Backtrace Summary Screen at the last executed statement, as shown in the following screen:

```

CA InterTest for CICS - BACKTRACE SUMMARY
Program= COBDEMO                               From 0005 To 0017 Of 0017
Specify S then ENTER to display Source Listing BACKTRACE
-----
PFKS 1 Help      2 Backtrace 3 End        4              5 1st Stmt  6 Last Stmt
      7 Backward 8 Forward   9 Next Wnd  10             11 Prev Bloc 12 Next Bloc
-----
S Bkmk      Stmt Block | Source Listing
-----
- ----      #375.0...#378.0| Call 'DFHEI1' using by content x'0204c000072c0
- ----      #378.0...#378.0| Go to WRITE-TSQ GEN-ERR depending on dfheigd
- ----      #407.0...#407.0| Call 'DFHEI1' using by content x'02048000071b0

```

```

- ---- #407.0...#410.0|      Call 'DFHEI1' using by content x'02048000071b0
- ---- #416.0...#416.0|      Call 'DFHEI1' using by content x'0e06c00007000
- ---- #416.0...#426.0|      Call 'DFHEI1' using by content x'0e06c00007000
- ---- #426.0...#432.0|      Call 'DFHEI1' using by content x'0e0a800007000
- ---- #432.0...#435.0|      Call 'DFHEI1' using by content x'02048000071b0
- ---- #438.0...#447.0|      MOVE SPACE TO TASK-SWITCH.
- ---- #447.0...#460.0|      Call 'DFHEI1' using by content x'0a02e00007000
- ---- #460.0...#468.0|      Call 'DFHEI1' using by content x'1804f00007000
- ---- #468.0...#470.1|      Call 'DFHEI1' using by content x'0402000007000
* ---- #479.0... BKPT |      ADD +1 TO TASKNUM.
CAIN2987 Last Backtrace Stmt Block

```

Read the Backtrace Summary

The Backtrace Summary screen summarizes the program's execution. This screen displays the statement number and source of the first statement in each contiguous group of executed program statements.

- The order of the statements reflects the program's execution path
- The top statement is the first statement executed; the bottom statement is the most recently executed statement.

From the Backtrace Summary screen, you can do the following actions:

- Reposition the display to view statement-by-statement details of the program's execution path by:
 - Pressing PF2
 - Marking a statement block with *s*, then pressing Enter
- Assign a *bookmark*, consisting of one to four characters, to one or more backtrace positions for faster and easier navigation through the Backtrace Summary
- Trace program execution using the following PF keys:
 - **PF5 1st Stmt**
Repositions the Backtrace Summary screen to the 1st or oldest entry in the Backtrace.
 - **PF6 Last Stmt**
Repositions the Backtrace Summary screen to the last or newest entry in the Backtrace.
 - **PF7 Backward**
Scrolls the Backtrace Summary screen backward (up) a full screen.
 - **PF8 Forward**
Scrolls the Backtrace Summary screen forward (down) a full screen.
 - **PF11 Prev Bloc**
Displays the previous statement block.
 - **PF12 Next Bloc**
Displays the next statement block.

Access the Source Listing Backtrace

To display the Source Listing Backtrace screen, press PF2 or type **s** next to a statement block in the Select column of the Backtrace Summary screen, as shown in the previous screen. The following screen is the Source Listing Backtrace screen.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BACKTRACE
COMMAND ==>
Program= COBDEMO  Option #      Stmt #      Search=      Margin= 01
-----
TASKNUM          | ?00000.
===== Backtrace at #000407 (000407 -> 000407 executed 1 times) <=====
000405*          PGMIDERR(NO-OPTS)
000406*          END-EXEC.
- 000407          Call 'DFHEI1' using by content x'02048000071b0000000000000000
- 000408-         '0000000000000000f0f0f1f7f6404040' end-call
- 000409          Service label
- 000410          Go to NO-OPTS depending on dfheigdi.
- 000411
000412*EXEC CICS LOAD
000413*          PROGRAM('IN250PTS')
000414*          SET(ADDRESS OF IN250PTS-AREA)
000415*          END-EXEC.
- 000416          Call 'DFHEI1' using by content x'0e06c0000700000400f0f1f7f9
- 000417-         '404040' by content 'IN250PTS' by reference ADDRESS OF
- 000418          IN250PTS-AREA end-call.
- 000419
- 000420          MOVE OPTS-CNTL TO TASK-CNTL.
- 000421          MOVE OPTS-PROTCPF TO TASK-PROTCPF.

```

Read the Source Listing Backtrace

To clearly detail the program's execution path, statement by statement, source statements that were executed from the current backtrace position are highlighted. When you specify a non-backtrace source listing option or use PF7 or PF8 to reposition the source listing, the highlighting of executed statements is temporarily suspended until a backtrace-related PF key (next block, previous block, and so on) is entered.

When in source listing backtrace mode, the current backtrace position always appears on the line just above the first source statement. This line displays as follows:

```
=====>BACKTRACE AT #nnnnn (nnnnn -> nnnnn executed nnnnn times)<=====
```

The information in this line indicates the following:

- **#nnnnn**
Specifies the program statement or offset the backtrace is currently positioned at
- **nnnnn -> nnnnn**
Specifies the first and last number of the statement block
- **nnnnn times**
Specifies how many times a statement block was executed

On the Source Listing Backtrace screen, each relevant backtrace position is indicated in the listing by an arrow or a line:

- **====>**
Indicates the first statement in a statement block

- <====
Indicates the last statement in a statement block
- ====
Indicates all other statements within a statement block

The Source Listing Backtrace screen displays the source of each statement:

- The order of the statements reflects the program's execution path.
- The top statement is the first statement executed; the bottom statement is the most recently executed statement.

Navigate through Program Execution

You can trace your program's execution path by using the following commands:

- Using PF5, PF9 and PF11 to trace execution backward
- Using PF6, PF10, PF12 to trace execution forward
- Setting the source listing profile Stepping Amount, and then using PF9 and PF10 to automatically trace the execution backward or forward

End a Source Listing Backtrace Session

To turn off the source listing backtrace display mode, press Clear or PF3. CA InterTest redisplay the Source Listing Breakpoint screen, positioned at the last breakpoint.

To toggle between the Source Listing Backtrace screen and the Backtrace Summary screen, press PF2.

Set the Code Counting Coverage Option

CA InterTest has many options for specialized testing needs. The code coverage option lets you see the number of times an assembler statement was executed.

The initial setting for code counting is OFF, which is the default. However, you can use the Source Listing Profile screen to turn it ON or OFF again at any time after you have selected monitoring for a program.

How the counts are displayed or not displayed is handled by the primary line command COUNTS and its associated parameters.

This feature causes overhead during program monitoring and should be turned off as soon as it is no longer needed.

To change the setting for code coverage, perform the following steps:

1. Type PROFILE on the command line and press Enter, or press PF4 to access the Profile screen from any Source Listing screen. The Code Counting= field shows the current setting.
2. Overtyping the current value with one of the following values:

- **YES**
Enables code counting and turns on the COUNTER display feature.
- **NO**
Removes the COUNTER display and stops the counting feature.

3. Press Enter to process the new settings and to return to the Source Listing Display screen.

4. The following screen details the code coverage feature:

```

CA InterTest for CICS - PROTSYM FILE  ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= COBDEMO  Option #          Stmt #                               Margin= 01
                               Search=
-----
----- TASKNUM                    | ?00000.
-----+-----
- 000351 PROCEDURE DIVISION using dfheiblk dfhcommarea.                | COUNTER | --
- 000352     MOVE EIBTRNID TO TSQ-TRANID.                               | 0000001 |
- 000353     MOVE EIBTRMID TO TSQ-TERMID.                               | 0000001 |
- 000354*** THIS CODE INITIALIZES THE MAPS BECAUSE COBOL2 DOESN'T
- 000355     MOVE LOW-VALUES TO DMAP04AI                                | 0000001 |
- 000356         DMAPBEGI
- 000357         DMAPASRI
- 000358         DMAPSUMI
- 000359         DMAPENDI
- 000360         DMAP00I
- 000361         DMAP01I
- 000362         DMAP02I
- 000363         DMAP03I
- 000364         DMAP04I
- 000365         DMAP05I
- 000366         DMAP06I
- 000367         DMAP07I

```

Statement Trace Facility

The Statement Trace Facility lets you view the logic flow of your program at a statement level. When used in conjunction with the Datamon command, it allows you to view the values of data items at the time that the statement was executed.



Note: This feature is only valid for COBOL programs. Use the Backtrace facility for tracing information for other languages.

Enable Statement Tracing and Data Monitoring

Use the TRACE command to enable the statement tracing facility. This creates a table whose number of entries is defined by the STMTTRCE option in IN25OPTS. This is a *wraparound* table, meaning that when all of the entries are used, the oldest entry is used for the current statement.

Use the DATAMON command, or DM, to enable data monitoring for the given program. If the Statement Trace facility was not in effect when data monitoring is set, a statement trace table is created.

Navigate the Statement Trace Table

When at a breakpoint, and statement tracing is in effect, you can use the PREV and ADVANCE commands to determine the execution path of the program prior to this point. Use the PREV command to cause the display to back up one statement. Consider the following breakpoint:

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= COBDEMO  Option #          Stmt #          Search=          Margin= 01
-----
----- TASKNUM | ?00000.
-----+-----
- 000477 CONTINUE-TASK.
- 000478**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
A ==> ADD +1 TO TASKNUM.
  ==>
  ==> ASRA ABEND (0C7) detected and prevented. Caused by invalid decimal
  ==> arithmetic data format.
  ==>
  ==> Press PF1 for a detailed description.
  ==>
- 000480 IF TASKNUM = 1
- 000481 MOVE 'DMAPASR' TO MAPNAME.
- 000482 IF TASKNUM = 2
- 000483 MOVE 'DMAPSUM' TO MAPNAME.
- 000484 IF TASKNUM GREATER 2
- 000485 GO TO SEND-END-MSG.
- 000486 GO TO REWRITE-TSQ.
- 000487 REWRITE-TSQ.

```

Using the PREV command on the previous screen results in the next screen:

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= COBDEMO  Option #          Stmt #          Search=          Margin= 01
-----
----- TASKNUM | ?00000.
-----+-----
- 000468 Call 'DFHEI1' using by content x'0402000007000000140000400000
- 000469- '00f0f0f2f0f9404040' end-call.
- 000470 IF EIBAID = DFHENTER GO TO CONTINUE-TASK.
- 000471 IF EIBAID = DFHCLEAR GO TO SEND-END-MSG.
- 000472 IF EIBAID = DFHPP3 GO TO SEND-END-MSG.
- 000473 IF EIBAID = DFHPP15 GO TO SEND-END-MSG.
- 000474 IF EIBAID = DFHPP2 GO TO EXPANDED-DEMO.
- 000475 IF EIBAID = DFHPP14 GO TO EXPANDED-DEMO.
- 000476 GO TO SEND-FIRST-SCREEN.
- 000477 CONTINUE-TASK.
- 000478**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
A ==> ADD +1 TO TASKNUM.
  ==>
- 000480 IF TASKNUM = 1
- 000481 MOVE 'DMAPASR' TO MAPNAME.
- 000482 IF TASKNUM = 2
- 000483 MOVE 'DMAPSUM' TO MAPNAME.

```

Statement number 470 was the statement executed just before the abend at the statement number 479. Similarly the ADVANCE command advances the statement trace pointer.

View Past Data Values

When data monitoring is enabled, the user is able to view past data values when navigating the trace table. Using the PREV and ADVANCE commands, data item values reflect the values from when that statement was executed. These values are protected -- you cannot update data items while viewing past data values.

The DATAMON command can be very CPU intensive -- it requires a significant amount of storage. Therefore, this command should be used with caution.

Indirect Commands

The CA InterTest Indirect Commands facility lets you specify a set of indirect commands to be executed in one of two ways:

- Automatically, when a breakpoint pointing to that set of commands is reached
- On demand, when you are at a breakpoint and use the Breakpoint Primary Option menu to resume execution with an indirect command (the fastpath is =4.5 from any breakpoint)

By using indirect commands, you can dynamically modify or correct programs without having to leave your test session to recompile your program.

With the indirect commands facility, you can do the following actions:

- Change the flow of control in your program
- Test conditions based on specified variables
- Change the value of specified variables
- Create and execute commands as a group, like adding a new subroutine
- Automatically resume execution of your program at the same or different location
- Define abbreviations for variables with long names

Code Indirect Commands

CA InterTest provides an indirect commands facility for coding the statements. You can access it from the source listing or from the ITST menus.

1. To access the indirect commands facility from the Source Listing Display screen, type ICMD5 in the command line or 11 in the Option # field and press Enter, as shown in the following screen :

```
CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==> icmd5
Program= COBDEMO  Option #          Stmt #          Search=          Margin= 01
-----+-----
00001 ID DIVISION.
00002 PROGRAM-ID. COBDEMO.
00003 ENVIRONMENT DIVISION.
00004 DATA DIVISION.
```

```

_ 00005 WORKING-STORAGE SECTION.
_ 00006 77 S999-FIELD1          PIC S9(3) .
_ 00007 77 S999-FIELD2          PIC S9(3)      VALUE +50.
_ 00008 77 999-FIELD1           PIC 9(3) .
_ 00009 77 999-FIELD2           PIC 9(3)      VALUE 50.
    
```

To access the Indirect Commands facility from the menus, access the Program Monitoring menu (fastpath is ITST 2.1), and select the Commands option. Perform the following steps:

1. Select option 2 - Monitoring on the Primary Option Menu. CA InterTest displays the Monitoring Menu.
2. Select option 1 - Programs. CA InterTest displays the Program Monitoring screen.
3. In the Program field, type the name of the program for which you want to create indirect commands.
4. Type an **s** to the left of the Commands option (Indirect commands), as shown in the following screen. Press Enter.

```

----- CA InterTest for CICS PROGRAM MONITORING -----
-----
COMMAND ==>

Type information and S to set or R to remove option(s) below.

Program . . cobdemo_      Program name (or .ALL, .OPTIONS or generic)
User ID . . _____    User (or .ANY) for whom the program is monitored

Option      Description                                          More:
+
_ Status    Display and/or remove monitoring options (S only)
_ Monitor   Monitoring (R removes monitoring and all options previously set)
_ UBP       Unconditional breakpoints (specific program only)
_ CBP       Conditional breakpoints (specific program only)
_ RBP       Breakpoints for CICS, DB2, DL/I or external CALL requests
_ Stmt Trace Statement tracing and data monitoring (COBOL only)
_ New copy  Fetch new copy of program and reset monitoring options (S only)
s Commands  Indirect commands defined for a specific COBOL or PL/1 program
_ Replace   CICS resource name replacement options
_ Protect   Storage protection monitoring options
_ Special   Other options (storage allocation, file updating, etc.)
_ Composite Monitor multi-CSECT program's separately compiled components

PF1 Help    2          3 End        4 Return    5          6
PF7 Backward 8 Forward  9          10         11         12
    
```

Either access method takes you to the Indirect Commands screen shown in the following example.

This example shows the indirect commands that you use to fix the ASRA condition in our COBDEMO demo program, without recompiling. Setting TASKNUM to 1 alters the demo results.

Use the clear key to complete the termination of the demo when the "You have completed the sample CA InterTest test session" screen appears.

```

CA InterTest MONITOR COMMAND BUILDER - COBDEMO INDIRECT COMMANDS 24
Term ID (or .ANY or .NO): U039 Delete ALL: NO

LINE COMMAND
-----
00010 /* INITIALIZE TASKNUM */
00020 MOVE 1 TO TASKNUM
00030 EXIT
    
```

```

1 Help      2          3 End      4          5          6
7 Backward 8 Forward 9 Top     10 Bottom 11         12
CAIN1261 Indirect command(s) added.

```

To code the indirect commands for your own programs, perform the following steps:

1. Specify the terminal ID (terminal ID, .ANY or .NO) where the indirect commands will take effect. The default terminal varies according to how you are monitoring the program.
 - If you are monitoring under a User ID of .ANY, the default terminal is the terminal you are using to specify the indirect commands.
 - If you are monitoring under your own User ID, the default terminal is .ANY.
2. Type the statement numbers and the text of the indirect commands as shown previously on the Indirect Commands screen. For more information, see Indirect Commands in [Breakpoint Activities \(https://docops.ca.com/display/CAITSD11/Breakpoint+Activities\)](https://docops.ca.com/display/CAITSD11/Breakpoint+Activities) and Online Help.
3. After coding the statements, press PF3 to exit the Indirect Commands facility. If you accessed the facility from the ITST menus, use =X to exit the menus.

How to Control the Flow of Execution

When you run your program, if a breakpoint points to indirect commands, the commands are automatically executed without stopping first. Execution is returned to the program according to the exit command that you specified in your indirect commands.

To return execution to your program following indirect command processing, use one of the following exit commands:

- **BREAK**
CA InterTest halts indirect command processing and issues a breakpoint display from where the indirect commands were invoked in the program.
- **GOTO**
CA InterTest resumes execution at the program statement number, offset, paragraph name, or indirect command statement number specified after the GOTO command.
- **EXIT**
CA InterTest resumes program execution at the breakpoint location from which the indirect commands were invoked and continues debugging the program until the next breakpoint.
- **RUN**
CA InterTest resumes program execution at the breakpoint location from which the indirect commands were invoked and ignores all subsequent breakpoints.

Attach the Indirect Command to Breakpoints

To have the indirect commands automatically executed by your program, set an unconditional, conditional, or request breakpoint that points to the first indirect command statement number. When you set a breakpoint with this option, each time the breakpoint takes effect the indirect commands are executed.

From the Source Listing Display screen, set the unconditional or conditional breakpoint as follows:

1. Display the line in your source code where you want to set the breakpoint to execute the indirect commands.
2. Type BPO in the command line or 12 in the Option # field (Breakpoint Options), tab down to the line where you want to set the breakpoint, type U for unconditional or C for conditional, and press Enter.
The Breakpoint Locations menu displays, which lets you set a number of breakpoint options.
3. Tab to the following option near the bottom of the panel:

Statement no. of indirect command(s) to be executed: _____

and type the first indirect command statement number (leading zeroes are optional), as shown in the following screen.

```
CA InterTest MONITORING COMMAND BUILDER - COBOL BREAKPOINT LOCATIONS      11
SET breakpoint options for  PROG=COBDEMO  for location:
000479 Execute indirect command 100
```

After: _

```
Enter 'n' to stop only every n'th time:
Term ID (or .ANY or .NO) where breakpoints will take effect:  U080
Term ID (or .ANY) that will receive the breakpoints:          U080
Statement no. of indirect command(s) to be executed:         10___
User ID (or .ANY) who will execute the program:               .ANY
```

PF1 Help	2	3 End	4 Return	5	6
PF7	8	9	10	11	12

4. Press Enter to process this menu.
5. If you are setting a conditional breakpoint, enter the condition on the Conditional Breakpoint screen and press Enter.
Your source listing display shows the indirect command to be executed on the line above the statement with the U or C indicator.

To set a request breakpoint that executes an indirect command, follow the usual steps to set the request breakpoint from the menus. On the Request Breakpoint Selection menu, also type the statement number of the indirect command to execute, as shown in the following screen.

```
CA InterTest MONITORING COMMAND BUILDER - REQUEST BREAKPOINT SELECTION  13
```

Set one or more types of Request Breakpoints in:
PROG=COBDEMO

<input type="checkbox"/> ALL commands	<input type="checkbox"/> DL/I	<input type="checkbox"/> DB2	<input type="checkbox"/> CALLs
<input type="checkbox"/> Address, Assign, Handles, Push, Pop	<input type="checkbox"/> Storage Control	<input type="checkbox"/> BMS	
<input type="checkbox"/> Terminal Control	<input type="checkbox"/> Program Control	<input type="checkbox"/> Trace Control	
<input checked="" type="checkbox"/> File Control	<input type="checkbox"/> Interval Control	<input type="checkbox"/> Dump Control	
<input type="checkbox"/> TD Control	<input type="checkbox"/> Task Control	<input type="checkbox"/> Batch Data Interchange	
<input type="checkbox"/> TS Control	<input type="checkbox"/> Journal Control	<input type="checkbox"/> Built-In Functions	
<input type="checkbox"/> Web access	<input type="checkbox"/> Syncpoints	<input type="checkbox"/> Sys Prog Functions	
	<input type="checkbox"/> Business Trans	<input type="checkbox"/> 3270 Bridge	

```

Enter 'n' to stop only every n'th time:      ----
Term ID (or .ANY or .NO) where breakpoints will take effect:  ----
Term ID (or .ANY) that will receive the breakpoints:          ----
Statement no. of indirect command(s) to be executed:         0010_
User ID (or .ANY) who will execute the program:              .any____

```

```

PF1 Help      2          3 End      4 Return    5          6
PF7           8          9         10         11         12

```

Review, Change or Delete Indirect Commands

To change, add, delete or review indirect commands directly from the Source Listing Display screen, type ICMDS in the command line or 11 in the Option # field and press Enter. CA InterTest displays the indirect commands for the default terminal, which is either the terminal you are working at or the terminal specified on the source listing profile.

For more information on the Indirect Commands facility, see [Breakpoint Activities \(https://docops.ca.com/display/CAITSD11/Breakpoint+Activities\)](https://docops.ca.com/display/CAITSD11/Breakpoint+Activities).

Replacement, Protection, and Special Options

CA InterTest has many options for specialized testing needs. Just a few are explained here; for more information on all the monitoring options, see [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging).

You can set or remove options from the Monitoring Command Builder menus. For more information, see [M \(see page 123\)onitoring Menu Options \(https://docops.ca.com/display/CAITSD11/Monitoring+Menu+Options\)](https://docops.ca.com/display/CAITSD11/Monitoring+Menu+Options).

Replacement Options

Replacement options allow you to dynamically change the names of CICS resources such as programs, files, transient data queues, and temporary storage.

One reason to use replacement options is to correct errors so you can continue testing without recompiling. For example, suppose CA InterTest halts your program at an automatic breakpoint to prevent an AEIL abend caused by an incorrect file name. You can use the Replace File Name option to dynamically change the file name so testing can continue.

Another reason for using replacement options is to use different resources during testing than those hard-coded in the program. For example, you can specify that the program use a test file rather than the production file named in the program.

Protection Options

When CA InterTest monitors a program, it prevents the program from modifying storage areas it does not own and issuing non-CICS instructions. Protection options let you override the CA InterTest default rules for modifying main storage, the CSA, and load modules. You can do the following procedures with these options:

- Modify any area of main storage
- Modify areas in the CSA or CWA

- Modify a load module

These options also prevent a program from modifying specific areas of storage not normally protected.

The protection options, which are password protected, provide extra flexibility for special testing needs.

Special Options

Special options alter the CA InterTest standard monitoring procedures and help you design customized testing scenarios.

The No File Updating option prevents a monitored program from updating files. This option is in testing because your file data is unchanged at the end of program execution. This means:

- You can repeatedly test a program without restoring your files
- Programs can share the same file without interfering with each other's work
- A test program can use a production file without affecting data integrity

Other options provide additional functions. For example:

- The FOL option lets CA InterTest continue monitoring even after a program branches directly to another program (wild branch)
- The MXS option limits the amount of CICS storage a program can use
- The MXR option limits the number of CICS requests a program can issue

Composite Module Testing

Composite support lets you take advantage of all CA InterTest functions when you test composite modules. A *composite module* consists of separately compiled modules link-edited in one load module. These modules may be written in different languages and compiled at different times. Composite support lets you test and debug a called subroutine as if it were a separate program with full symbolic support.

Example

Suppose a COBOL program has several subroutines, some written in COBOL, some written in Assembler and PL/I. You can test and debug any of the subroutines just as if it were a main program by setting breakpoints, setting monitoring options, displaying the source listing, and displaying and modifying main and auxiliary storage.

Composite module monitoring is initiated by specifying the COMPOSITE command in the Source Listing Display screen before monitoring is activated (PF5) for the program.

```
CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==> COMPOSITE
Program= COBDEML  Option #          Stmt #          Margin= 01
                   Search=
-----
00004 DATA DIVISION.
```

```

_ 00005 WORKING-STORAGE SECTION.
_ 00006 77 NUM-CHOICES          PIC S9(4) COMP VALUE +2.
_ 00007 01 DFHAID COPY DFHAID.
_ 00008*
_ 00009*   COPYRIGHT = 5655-018 (C) COPYRIGHT IBM CORP. 1985 =
_ 00010*   THIS MODULE IS "RESTRICTED MATERIALS OF=IBM"
_ 00011*   LICENSED MATERIALS - PROPERTY OF IBM =
_ 00012*   REFER TO COPYRIGHT INSTRUCTIONS =
_ 00013*   FORM NUMBER G120-2083 =
_ 00014*
_ 00015 01 DFHAID.
_ 00016 02 DFHNULL PIC X VALUE IS ' '.
_ 00017 02 DFHENTER PIC X VALUE IS ' '.
_ 00018 02 DFHCLEAR PIC X VALUE IS ' '.
_ 00019 02 DFHCLRP PIC X VALUE IS ' '.
    
```

When you specify the COMPOSITE command, CA InterTest displays the link-edit information for the main program and its subroutines. You can change this information online, as shown in the following screen:

```

          CA InterTest for CICS - Composite Support Builder
COMMAND ==>                                     SCROLL: PAGE
Define composite support for COBDEML and press PF5.      Row 00001 of 00003
    
```

* * * * *	* * * * *	* * * * *	* * * * *	* * * * *	* * * * *
Link name	Monitor	Offset	Length	Language	Comments
S COBDEML	COBDEML	20	748	IBMCOB 4.2	
S ASBIN25	ASBIN25	768	17C	HLASM 6.0	
S CSBIN25	CSBIN25	8E8	674	IBMCOB 4.2	

```

PF1 Help      2          3 End          4          5 Process    6 RFind
PF7 Backward  8 Forward  9          10         11          12
    
```

Press PF5 to enable composite module monitoring so each program can be tested separately.

Dump Analysis with CA SymDump

CA SymDump for CICS -- the CA InterTest companion symbolic dump facility for z/OS users -- is a powerful online tool that automatically pinpoints the source statement responsible for an abend. CA SymDump was designed to complement CA InterTest, especially in production regions where programs are not usually monitored. Working with CA InterTest for CICS, CA SymDump for CICS lets you bring a dump back to life. Familiar CA InterTest source listing screens and other facilities, such as the ability to view and modify main and auxiliary storage, make it easy to resolve application dumps.

CA SymDump is option 5 - Dump analysis on the CA InterTest Primary Option Menu.

With CA SymDump, you can do the following actions:

- Analyze dumps symbolically online, using CA InterTest source code listings

- Resolve CICS production dumps from any region; for example, debug production abends from your test region
- Manage your dump data set, selectively retaining the dumps you want to view and print, and discarding ones you do not need

When you use CA SymDump to view a dump, you can display the following items:

- CA InterTest breakpoint at the instruction that triggered the abend
- Formatted system areas
- Formatted trace table
- Registers and displacement at the abend

Example

Suppose a program not being monitored by CA InterTest for CICS abended with an ASRA. With CA SymDump for CICS, you can display the breakpoint at the statement that triggered the abend even though CA InterTest for CICS was not active in the region when the abend occurred.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED DUMP ANALYSIS
COMMAND ==>>
Program= COBDEMO  Option #          Stmt #          Displacement=          Margin= 01
                  Search=
-----
- 000477 CONTINUE-TASK.
- 000478**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
A ==>>      ADD +1 TO TASKNUM.
  ==>>
  ==>> ABEND/DUMP CODE: ASRA
  ==>>
  ==>>      Press PF1 for a detailed description.
  ==>>
- 000480      IF TASKNUM = 1
- 000481          MOVE 'DMAPASR' TO MAPNAME.
- 000482      IF TASKNUM = 2
- 000483          MOVE 'DMAPSUM' TO MAPNAME.
- 000484      IF TASKNUM GREATER 2
- 000485          GO TO SEND-END-MSG.
- 000486      GO TO REWRITE-TSQ.
- 000487 REWRITE-TSQ.
- 000488*EXEC CICS WRITEQ TS
000489*      REWRITE
000490*      QUEUE(TSQ-NAME)

```

Now you can use CA SymDump to solve the problem. In this example, you might want to display the contents of TASKNUM, or you might want to inspect a file or look at the formatted trace table. CA SymDump provides all the tools you need to diagnose and resolve the dump.

COBOL Advanced Demo Session

This article contains some of the advanced features of CA InterTest for CICS. For more information, see [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging) or the Help facility.

- [Demo Preliminaries \(see page 142\)](#)
- [Execute the Demo Program \(see page 148\)](#)
- [Option 01 Replace a File Control ID \(see page 149\)](#)
- [Option 02 Limit CICS Storage and Requests \(see page 153\)](#)
- [Option 03 Prevent a Program from Updating a File \(see page 157\)](#)
- [Option 04 How to Display Variable Length Data \(see page 163\)](#)
- [Option 05 How to Work with Indexed Table Items \(see page 168\)](#)
- [Option 06 How to Detect a Storage Violation \(see page 173\)](#)
- [Option 07 How to Test a Composite Module \(see page 176\)](#)

If you are a new user, you might want to postpone this advanced demo session until you are more comfortable with the basic features of CA InterTest for CICS.

This article consists of the following independent sections. Each details a different advanced CA InterTest for CICS feature:

- **Option 01**
Replace a file name.
- **Option 02**
Limit the number of CICS requests issued by a program.
Limit the amount of storage obtained by a program.
- **Option 03**
Prevent a program from updating a file.
Use request breakpoints and the FILE facility.
- **Option 04**
Display variable length data.
- **Option 05**
Display indexed COBOL table items.
- **Option 06**
Detect and prevent a storage violation.
- **Option 07**
Test a composite module.

Before proceeding with the demo session, you must complete the steps outlined in the next section.

Demo Preliminaries

The advanced demo session uses COBDEMO just as the basic session did. Before beginning, you must set some breakpoints that will be used in different sections of the advanced demo session.

Set Unconditional Breakpoints

To begin, you must bring up the Source Listing display for the COBOL demo program. This was covered in the basic demo session.

1. Press PF4 to display the Profile panel. Verify that the Display window is set to "N" and that the AutoKeep Display is "ON". If not, change either or both of them and press Enter.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING PROFILE
COMMAND ==>
Program= COBDEMO Option # Stmt # Margin= 01
Search=
OPTS 1 Proc div 2 Work-stor 3 Link sect 4 D-map 5 Clst/Pmap More: +
      6 Data xref 7 Proc xref 8 Err msgs 9 Srch fwd 10 Srch bwd
PFKS 1 Help 2 3 End 4 Auto prms 5 Monitor 6 Menu
      7 Backward 8 Forward 9 Next Wnd 10 11 12 Status
-----
Display window = N N (None), T (Titles), R (Registers),
                 K (Keep), P (Program)
PF7/8 amount = PAGE PAGE, HALF, STOP, or a number from 1 to 9999
Step Timing = BEFORE Stop Before or After the next verb is executed
Stepping amount = 001 The number of verbs to execute
Auto-stepping = OFF ON to activate; press PF4 to change values
Source List BKPT = ON OFF to use the detailed breakpoint display
From terminal ID = U049 Terminal ID where the program will execute
BKPT terminal ID = U049 Terminal ID to receive the breakpoint displays
User ID = .ANY User ID who will execute this program
AutoKeep Display = ON OFF to deactivate
Code Counting = OFF ON to activate Code Coverage
SDF = DATA HEX for Hexadecimal/Character Format
    
```

2. Press PF3 to return to the listing.
Now you are going to set unconditional breakpoints at three procedure names. Setting unconditional breakpoints lets you halt the program so that you can set options and check the contents of data items.

The quickest way to set breakpoints at *several* procedure names is to display a list of all procedure names, and type **U** or **)** next to the ones where you want breakpoints. Because a procedure name does not have executable code, the unconditional breakpoint halts execution before or after the first verb in the paragraph. The **U** sets a breakpoint that stops *before* the first verb. The **)** sets a breakpoint that stops *after* the first verb.

3. Type **L.PX** in the command line or 7 in the Option # field, and press Enter.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==> l .px
Program= COBDEMO Option # Stmt # Search=
-----
00001 ID DIVISION.
00002 PROGRAM-ID. COBDEMO.
    
```

CA InterTest for CICS displays a list of the demo program's procedure names, as shown in the following figure.

You are going to set unconditional "before" breakpoints at three procedure names: AFTER-REWRITE, DO-READ-VAR, and LOOP-RTN.

4. Type **U** to the left of AFTER-REWRITE, DO-READ-VAR, and LOOP-RTN.

5. Press Enter.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>
Program= COBDEMO Option # Stmt # Margin= 01
Search=
-----
- DEFINED PROCEDURE NAMES REFERENCES
u 923 AFTER-REWRITE P798
- 800 CICS-LOOP G402 G470
- 477 CONTINUE-TASK
    
```

```

-      667 DATA-NAME
u      686 DO-READ-VAR                G683
-      555 EXPANDED-DEMO             G399 G474 G475
-      884 GEN-ERR                    D378 D435 D633
-      821 GETMAIN-LOOP              P819
-      661 INITIALIZE-TABLE          P648
-      538 LAST-SCREEN                D530
-      982 LINK-DEML                 D979
u      797 LOOP-RTN
-      737 MOVE-RECORD
-      792 MXR-OPTION                D974
-      806 MXS-OPTION                G799
-      437 NO-OPTS                    D410
-      830 NO-STORAGE                D817
-      777 NOT-OPEN                  D708
    
```

CA InterTest for CICS sets the breakpoints and redisplay the screen with uppercase Us.

Set Request Breakpoints

Request breakpoints make it easy for you to halt your program at various points so you can check program processing. For example, you can halt the program before every READ command, single-step through the read instructions, and then check main storage to make sure the program has read the correct record.

You can set request breakpoints to halt a program before CICS commands and macros and other calls, such as calls to DL/I or DB2. You can set request breakpoints prior to all CICS commands (as you might have done with IBM's CEDF) or prior to specific commands, such as File Control or Program Control commands.

For option 03 of the advanced demo session, you need to set request breakpoints at all File Control REWRITE commands in the demo program. To do this, do the following:

1. Specify the RBP command on the command line.

CA InterTest for CICS displays the Request Breakpoint Selection menu on which you specify the type of request breakpoint, such as: all CICS commands, File Control and Task Control.

CA InterTest MONITORING COMMAND BUILDER - REQUEST BREAKPOINT SELECTION 13

```

          Set one or more types of Request Breakpoints in:
          PROG=COBDEMO

- ALL commands          - DL/I    - DB2    - CALLs
- Address, Assign,     - Storage Control    - BMS
  Handles, Push, Pop   - Program Control    - Trace Control
- Terminal Control     - Interval Control   - Dump Control
s File Control         - Task Control       - Batch Data Interchange
- TD Control          - Journal Control    - Built-In Functions
- TS Control          - Syncpoints         - Sys Prog Functions
- Web access          - Business Trans     - 3270 Bridge

Enter 'n' to stop only every n'th time:
Term ID (or .ANY or .NO) where breakpoints will take effect:
Term ID (or .ANY) that will receive the breakpoints:
Statement no. of indirect command(s) to be executed:
User ID (or .ANY) who will execute the program:

PF1 Help      2          3 End      4 Return   5          6
PF7           8          9         10        11        12
    
```

2. Tab to the File Control field and type S.

3. Press Enter.

CA InterTest for CICS displays the File Control Request Breakpoint Selection menu shown in the following screen.

CA InterTest MONITORING COMMAND BUILDER - REQUEST BREAKPOINT SELECTION

Set one or more types of File Control COMMANDS IN:
PROG=COBDEMO

_ All commands

```

- READ
- WRITE
- REWRITE
- DELETE
- UNLOCK
- STARTBR
- READNEXT
- READPREV
- ENDBR
- RESETBR

```

Enter 'n' to stop only every n'th time ----

PF1 Help	2	3 End	4 Return	5	6
PF7	8	9	10	11	12

On this menu you can select the specific File Control commands where you want to set request breakpoints. In this case select REWRITE to halt the program prior to each File Control REWRITE command.

4. Tab to the REWRITE command field.

5. Type S and press Enter.

CA InterTest MONITORING COMMAND BUILDER - REQUEST BREAKPOINT SELECTION

Set one or more types of File Control COMMANDS IN:
PROG=COBDEMO

_ All commands

```

- READ
- WRITE
S REWRITE  ç
- DELETE

```

CA InterTest for CICS sets request breakpoints prior to each File Control REWRITE command in the demo program and redisplay the Program Monitoring Menu.

The next topic is necessary only if you want to use the composite support option. Composite support is a CA InterTest for CICS feature designed to aid programmers who are responsible for testing and debugging called subroutines.

- If you do not want to use the composite support option, skip to [Execute the Demo Program \(see page 148\)](#).
- If you do want to use the composite support option, perform the steps in Composite Support that follows.

Composite Support

Composite support lets you use CA InterTest for CICS to test and debug composite modules. A *composite module* is a load module, defined to CICS as a program, which consists of separately compiled or assembled parts brought together when the module is link-edited. In the demo session, the part of the composite module that receives control from CICS is referred to as the *main program*; the remaining programs are referred to as *subroutines*. The main program and subroutines can be written in the same or different languages.

Composite support lets you test and debug a subroutine, such as an external procedure, as if it were a separate program with full symbolic support. This means you can set breakpoints and other monitoring options individually for any subroutine. CA InterTest for CICS generates automatic breakpoints in the subroutine when it detects an error that would otherwise cause the program to abend.

To demonstrate this feature, you set composite support for a composite module, to which the demo program links. The composite module that you use is COBDEML.

The composite module has two called subroutines that you want CA InterTest for CICS to monitor -- subroutine CSBIN25 written in COBOL, and subroutine ASBIN25 written in Assembler.

Follow these steps:

1. In the Program Monitoring Menu, press PF4 until the Source Listing Display is redisplayed.
2. In the Program field, overwrite the name of the demo program with the name of the composite module COBDEML, and press Enter.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>
Program= COBDEML Option #      Stmt #      Search=      Margin= 01
-----
YEARWINDOW(1900)
ZWB
000001 ID DIVISION.
000002 PROGRAM-ID. COBDEML.
000003 ENVIRONMENT DIVISION.
000004 DATA DIVISION.
000005 WORKING-STORAGE SECTION.
- 000006 77 NUM-CHOICES          PIC S9(4) COMP VALUE +2.
- 000007 COPY DFHAID.
- 000008*
000009*          @BANNER_START@
000010*          Licensed Materials - Property of IBM
000011*
000012*          "Restricted Materials of IBM"
000013*
000014*          5655-147
000015*
000016*          (C) Copyright IBM Corp. 1985
000017*

```

CA InterTest for CICS displays the source listing for the composite module.

3. Type COMPOSITE on the command line and press Enter.

```

CA InterTest for CICS - Composite Support Builder
COMMAND ==>
Define composite support for COBDEML and press PF5.
SCROLL: PAGE
Row 00001 of 00003

```

* *	*	*	*	*	*
Link name	Monitor	Offset	Length	Language	Comments
S COBDEML	COBDEML	20	748	IBMC0B 4.2	
S ASBIN25	ASBIN25	768	17C	HLASM 6.0	
S CSBIN25	CSBIN25	8E8	674	IBMC0B 4.2	

PF1 Help	2	3 End	4	5 Process	6 RFind
PF7 Backward	8 Forward	9	10	11	12

4. Press PF5 to activate composite program monitoring for COBDEML.

Check the Monitoring Status Display

Before continuing, we will check the Monitoring Status display for the program COBDEMO. You should see the options you set for the program, which are needed to complete the Demo Options in the rest of this article.

1. On the Program Monitoring menu, scroll the Options list to the beginning. Either enter the TOP command and press Enter or press PF7.

```

----- CA InterTest for CICS PROGRAM MONITORING -----
-----
COMMAND ==>

      Program . . cobdemo_      Program name (or .ALL, .OPTIONS or generic)
      User ID . . _____    User (or .ANY) for whom the program is monitored

      Option      Description                                     More: +
s Status        Display and/or remove monitoring options (S only)
_ Monitor       Monitoring (R removes monitoring and all options previously set)
_ UBP           Unconditional breakpoints (specific program only)
_ CBP          Conditional breakpoints (specific program only)
_ RBP          Breakpoints for CICS, DB2, DL/I or external CALL requests
_ Stmt Trace   Statement tracing and data monitoring (COBOL only)
_ New copy     Fetch new copy of program and reset monitoring options (S only)
_ Commands    Indirect commands defined for a specific COBOL or PL/1 program
_ Replace      CICS resource name replacement options
_ Protect     Storage protection monitoring options
_ Special     Other options (storage allocation, file updating, etc.)
_ Composite    Monitor multi-CSECT program's separately compiled components

PF1 Help      2          3 End          4 Return      5          6
PF7 Backward  8 Forward   9          10           11          12
    
```

2. Type an **s** next to the Status option to obtain a Monitoring Status display of the COBDEMO program.
The Monitoring Status display for COBDEMO should show the three breakpoints needed to perform the Demo Session Options; the Request Breakpoint (RBP) at all REWRITE commands, and the two Unconditional Breakpoints (UBPs).

From the Welcome Screen you can access the Options Menu. Each of the options is described in detail in the sections that follow.

4. Press PF2 to go to the Options Menu of the demo.
 The demo program displays the Options Menu, as shown next.

```
*****
****                                     ****
****                               CA InterTest Demo Session          ****
****                               Options Menu                       ****
****                                     ****
*****
```

```
01 Replace file control ID
02 Limit CICS storage and requests
03 No file updating
04 Display variable length data
05 Work with table items
06 Storage violation detection
07 Composite support
```

Key in request: 01

Press ENTER to continue or CLEAR to terminate.

Option 01 Replace a File Control ID

This section of the demo session illustrates how to replace a File Control ID. The demo program is attempting to read and display a record, but the file name specified in the program is incorrect, possibly because it was misspelled.



Note: Before proceeding, ensure you have completed the steps outlined in the [Demo Preliminaries](#) (see page 142) section, and that you have the CA InterTest Demo Session Options Menu displayed on your screen.

1. Select option 01 and press Enter.
 The demo program displays the following screen, which describes what occurs in this part of the demo session.

```
*****
****                                     ****
****                               CA InterTest Demo Session          ****
****                               Replace File Control ID Option      ****
****                                     ****
*****
```

The program attempts to read file PROTH. Because the file name has been incorrectly specified, this would cause a DSIDERR condition which would result in an AEIL abend. Instead, here is what will happen:

1. CA InterTest halts the program at an automatic breakpoint.
2. You set the Replace File Control option to change the file name without recompiling.

3. You re-execute the program from the point at which it first tried to read the file.

When this section of the demo completes, you will return to the Options Menu.

Press ENTER to continue or CLEAR to terminate.

2. Press Enter.
CA InterTest for CICS halts the demo program at an automatic breakpoint.

Prevent an AEIL Abend

The following screen shows that CA InterTest for CICS has halted the demo program at an automatic breakpoint before an AEIL abend could occur.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= COBDEMO  Option #          Stmt #          Search=          Margin= 01
-----
----- RECORD-KEY | 0000000000000000000000000000000000000000000000000000000000000000
-----+-----
000711*EXEC CICS STARTBR DATASET('PROTH') RIDFLD(RECORD-KEY)
000712*      END-EXEC.
A  ==>      Call 'DFHEI1' using by content x'060cb000070020a000f0f0f3f9f3
    ==>
    ==>      CICS abend intercepted. Abend code = AEIL. EXEC CICS request was
    ==>      STARTBR. Now stopped AFTER the CALL.
    ==>
    ==>      Press PF1 for a detailed description.
    ==>
_ 000714-    '404040' by content 'PROTH ' by content x'0000' by content
_ 000715    x'0000' by reference RECORD-KEY end-call.
_ 000716
_ 000717    MOVE +1 TO SUB.
_ 000718*EXEC CICS READNEXT DATASET('PROTH')
_ 000719*      SET(ADDRESS OF RECORD-AREA)
_ 000720*      LENGTH(REC-LEN) RIDFLD(RECORD-KEY) END-EXEC.
_ 000721    Call 'DFHEI1' using by content x'060ef400070100b000f0f0f3f9f6

```

The abend that CA InterTest for CICS prevented would have resulted from the EXEC CICS STARTBR command. CA InterTest for CICS highlighted the CALL statement that is part of the generated EXEC CICS STARTBR command.

What is the problem?

The data set name PROTH is wrong; it should be PROTHLF.

How can it be fixed?

CA InterTest for CICS lets you dynamically correct the file name and continue testing. The following section explains how this is done.

Change the File Name

CA InterTest for CICS lets you dynamically change an incorrect file name in the middle of your test session. This feature lets you continue testing without recompiling your program. In this case, you can correct a simple typo in the demo program without interrupting your test session or changing your File Control Table.

To change the file name, you need to access the Replacement Options menu.

1. Specify the RO command on the Source Listing Display command line.
CA InterTest displays the Replacement Options menu.
The Set Replacement Options menu lets you dynamically change a program name, file name, transient data queue name, or temporary storage ID.
2. Tab to the Replace file name field.
3. Specify the incorrect file name (proth) in the first column and the correct file name (prothlf) in the second column, and press Enter.

```

CA InterTest MONITORING COMMAND BUILDER - REPLACEMENT OPTIONS      20
SET one or more options to replace one CICS resource name with another in:
PROG=COBDEMO

Replace program name:          _____ with _____
Replace file name:             PROTH___ with PROTHLF_
Replace TD queue name:        _____ with _____
TS selection mask:            _____
TS replacement mask:          _____

Limit monitoring to your terminal - '*' or TERMIID:
User ID (or .ANY) who will execute the program:      .ANY

```

```

PF1 Help      2          3 End      4 Return    5          6
PF7           8          9         10         11         12

```

CA InterTest dynamically replaces the incorrect file name when the program executes. CA InterTest then redisplay the Program Monitoring menu.

4. Press PF4 until the Source Listing Breakpoint Screen is redisplayed.

Resume Execution

Now that you have dynamically corrected the file name, you are ready to resume program execution. However, be aware that when an error occurs in a paragraph containing a CICS command, you should back up to the paragraph-name rather than resume execution at the specific instruction (some CICS internal processing might have occurred prior to the breakpoint). In this case it is important to resume execution at READ-DATASET to ensure that the demo proceeds as expected.

1. On the source listing breakpoint screen, position the listing so the READ-DATASET label is displayed by pressing PF7, tab to READ-DATASET, and enter G (go).

- You can dynamically change a file name to test a program using a different file. This lets you use a test file without altering the name of the production file hard-coded in the program.
- You can change a program name or transient data queue name to test a program or queue other than the ones that are hard-coded.

This concludes Option 01 of the advanced demo session. Press Enter to continue with the demo session or press Clear to terminate the session.

Option 02 Limit CICS Storage and Requests

This section of the demo session shows how to limit CICS main storage and CICS requests for a specific program (the demo program).

Two CA InterTest for CICS options are detailed in this part of the demo session:

- The MXS option limits the amount of main storage obtained by a task. This option is useful in detecting program errors that cause a task to acquire an excessive amount of storage.
- The MXR option limits the number of CICS requests, commands and macros that a program can issue. This option is useful in detecting program loops that generate an excessive number of CICS requests.



Note: Before proceeding, be sure you have completed the steps outlined in the Demo Preliminaries section, and that you have the CA InterTest Demo Session Options Menu displayed on your screen.

1. Select option 02, and press Enter.

The demo program displays a screen that describes what occurs in this part of the demo session.

```
*****
****
****                      CA InterTest Demo Session          ****
****                      MXR Option - Limit Number of CICS Requests ****
****                      MXS Option - Limit Amount of CICS Storage ****
****
*****
```

The program has a loop containing a CICS request and another loop containing a GETMAIN request. Here is what will happen:

1. The program is halted at an unconditional breakpoint before the loops.
2. You set the MXS and MXR options to limit storage and CICS requests.
3. The program continues to execute.
4. CA InterTest halts the program at automatic breakpoints when the limits are exceeded.
5. You remove the options and the program completes execution.

When this section of the demo completes, you will return to the Options Menu.

Press ENTER to continue or CLEAR to terminate.

2. Press Enter.

CA InterTest for CICS halts the program at the procedure name LOOP-RTN (because you set an unconditional breakpoint there as part of the Demo Preliminaries). Because a procedure name does not have executable code, the breakpoint you set halts execution before the first verb in the paragraph.

In this case, CA InterTest for CICS halts the demo program at the highlighted PERFORM command. Now you can access the Special Options menu to set the options you need for testing.

Set the MXS and MXR Options

1. In the Command field, specify the SO command.

CA InterTest for CICS displays the Special Options menu. The Special Options menu lets you set several options to change how CA InterTest for CICS monitors a program.

You are going to set two options, as shown following:

- MXS, to limit the amount of CICS main storage acquired by the demo program
- MXR, to limit the total number of CICS requests issued by the demo program

2. Specify 50,000 bytes as the maximum amount of main storage the demo program obtains, and specify 20 as the maximum number of CICS requests the demo program issues.

3. Tab to the MXS field and type in 60000, as shown in the following screen.

4. Tab to the MXR field and type in 20, as shown in the following screen.

5. Press Enter.

```

CA InterTest MONITORING COMMAND BUILDER - SPECIAL OPTIONS          22
SET one or more options to override the default monitoring rules in:
      PROG=COBDEMO

Enter X next to each option desired:

      Source Listing Breakpoint      (SLB)  _
      No file updating               (NUP)  _
      Reentrancy check               (RNT)  _

Follow monitoring (ON, name, NOPPT) (FOL)  _____
Number of times to be monitored     (MUS)  _____
Limit total size of CICS storage     (MXS)  60000__
Limit total number of CICS requests (MXR)  20____
Structure Display Format (HEX,DATA)  (SDF)  ____

Set local automatic breakpoint ('*', TERMID, .ANY, OFF)  _____
Limit monitoring to your TERMINAL - '*' or TERMID:      _____
User ID (or .ANY) who will execute the program:         .ANY

PF1 Help      2          3 End      4 Return   5          6
PF7           8          9          10         11         12
    
```

CA InterTest for CICS limits main storage and CICS requests for the demo program and redisplay the Program Monitoring screen.

6. Press PF4 until you return to the Source Listing Breakpoint screen.
7. Press PF5 to continue program execution.
The demo program resumes execution.

Limit CICS Requests

The next screen you see is an automatic breakpoint display screen. CA InterTest for CICS has halted the demo program at an automatic breakpoint because the maximum number of CICS requests (20) has been exceeded.



Note: If you are working on a system with LE 370 runtimes, you might reach the automatic breakpoint for the MXS (max storage size) first. If you do, simply perform the demo in the following order:

1. Remove the MXS options by typing STATUS, in the command line then 'R' next to the MXS option, and hit enter. Resume execution.
2. After stopping for the MXR trigger value, remove it the same as the MXS and continue.
3. The demo will execute to completion.
The instruction that triggered the automatic breakpoint is within a loop containing a CICS request.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= COBDEMO  Option #          Stmt #          Search=          Margin= 01
-----
000801*EXEC CICS ASKTIME
000802*          END-EXEC.
A  ==>    Call 'DFHEI1' using by content x'100200000700001300f0f0f4f5f4
    ==>
    ==> MXR (max CICS requests) trigger value exceeded.
    ==>
    ==>    Press PF1 for a detailed description.
    ==>
_ 000804-  '404040' end-call.
  000805
  000806 MXS-
OPTION.

```

Remove the MXR Option

Now you can remove the MXR option so that the demo program continues executing. If this were your own program, you would examine the instructions that are generating excessive CICS requests.

1. Type STATUS on the command line and press Enter.
CA InterTest for CICS displays the Monitoring Status screen.

```

----- CA InterTest for CICS  MONITORING STATUS -----
COMMAND ==>

Type + to expand or - collapse option levels displayed below,
      or R to remove option(s).

```

More: +

Option	Description	Attributes
- COBDEMO	Program monitor entry	IBMC0B 3.2
	Waiting at breakpoint	Task 00156, ABP since 10:33 a.m.
- -.ANY	User monitoring options	Active
	Symbolic listing file	PROTSYM
- -MXR	Maximum CICS requests	20
- -MXS	Maximum total storage	50000
- -RBP	Request breakpoint(s)	REWRITE
	Option ID	ECEAD6CB
	From, to terminals	U061, U061
- -UBP	Unconditional breakpoint	'AFTER-REWRITE'
	Option ID	D87A3D68
	From, to terminals	U061, U061
- -UBP	Unconditional breakpoint	'LOOP-RTN'
	Option ID	6CD871EC

2. Remove the MXR option by typing an **r** next to the MXR breakpoint's entry and pressing Enter. CA InterTest for CICS removes the breakpoint.
3. Press PF3 to return to the Source Listing Breakpoint screen.
4. Press PF5 to resume program execution.

Limit Acquisition of Storage

The next screen you see is an automatic breakpoint display. CA InterTest for CICS halts the demo program at an automatic breakpoint because the maximum main storage limit of 60,000 has been exceeded. In most cases, the instruction that triggered the automatic breakpoint is within a loop, which contains a GETMAIN request.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= COBDEMO  Option #          Stmt #          Search=          Margin= 01
----- DFHB0020          | +06144.
-----+-----
000825*          END-EXEC.
- 000826          Move 6144 to dfhb0020
A ==>          Call 'DFHEI1' using by content x'0c02c0000700008c00f0f04f6f8
==>
==> MXS (max storage size) trigger value exceeded.
==>
==>          Press PF1 for a detailed description.
==>
- 000828-          '404040' by reference ADDRESS OF GETMAIN-AREA by reference
- 000829          dfhb0020 end-call.
- 000830 NO-STORAGE.
- 000831          MOVE 'B' TO TASK-SWITCH3.
- 000832          MOVE 'NO STORAGE' TO ERROR0.
- 000833*EXEC CICS SEND
000834*          MAP ('DERROR')
000835*          MAPSET ('IN25CMP')
000836*          ERASE
    
```

Remove the MXS Option

Now you can remove the MXS option so that the demo program continues executing. If this were your own program, you would examine the instructions that are causing the program to obtain excessive amounts of storage.

1. Type STATUS on the command line and press Enter. CA InterTest for CICS displays the Monitoring Status screen.

```

----- CA InterTest for CICS MONITORING STATUS -----
COMMAND ==>

Type + to expand or - collapse option levels displayed below
or R to remove option(s).

Option      Description      Attributes      More:  +
- - COBDEMO  Program monitor entry  COBOL
  |          Waiting at breakpoint  Task 00083, ABP since 03:50p.m.
- - |-.ANY   User monitoring options  Active
  |          Symbolic listing file    PROTSYM
r   | -MXS   Maximum total storage    50000
  
```

2. Remove the MXS option by typing an R next to the MXS breakpoint's entry and pressing Enter. CA InterTest for CICS removes the breakpoint.
3. Press PF3 to return to the Source Listing Breakpoint screen.

Demo Program Completes Execution

Press PF5 to resume program execution.
The demo program completes execution. CA InterTest for CICS returns to the demo's Options Menu.

Review What Happened

In this part of the test session you set two options to help test the demo program, as shown following:

- The MXR option lets you limit the number of CICS requests the program could issue. When the demo program exceeded this limit, CA InterTest for CICS halted the program. This feature helps you find program instructions within a loop that generate an excessive number of CICS requests.
- The MXS option let you limit the amount of main storage used by the program. When the demo program exceeded the amount of storage you specified, CA InterTest for CICS halted the program. This feature helps you find commands or macros within a loop that are obtaining excessive amounts of storage.

This concludes Option 02 of the advanced demo session. Press Enter to continue with the demo session or press Clear to terminate the session.

Option 03 Prevent a Program from Updating a File

This section of the demo session details how to execute a program without having it update a file.

The No File Updating option is useful when you are testing a program. Preventing a program from updating a file means your test data remains unchanged at the end of each program execution, so you can test the program repeatedly without having to recreate test data.

Many programs can use the same test file without interfering with each other's work. You can even allow a test program to use a production file because the integrity of the file is preserved.



Note: Before proceeding, be sure you have completed the steps outlined in [Demo Preliminaries \(see page 142\)](#) and that CA InterTest Demo Session Options Menu is displayed on your screen.

Also verify that the CA InterTest for CICS checkpoint file (PROTCPF) has been defined to your CICS region and that the file's current status is Open and Enabled.

1. Select option 03, and press Enter.

The demo program displays the following screen that describes what occurs in this part of the demo session.

```
*****
****                                     ****
****                               CA InterTest Demo Session          ****
****                               No File Updating Option            ****
****                                     ****
*****
```

The program reads a record and changes data in the record. Here is what will happen:

1. CA InterTest halts the program at a request breakpoint.
2. You set the No File Updating option to prevent COBDEMO from updating the file.
3. When COBDEMO rewrites the file, the file is not updated.
4. After the rewrite you use the FILE facility to confirm that the file has not been updated.

When this section of the demo completes, you will return to the Options Menu.

Press ENTER to continue or CLEAR to terminate.

2. Press Enter.

CA InterTest for CICS displays a request breakpoint screen.

Halt the Demo Program at a Request Breakpoint

CA InterTest for CICS has halted the demo program prior to the first File Control REWRITE command as you specified during the Demo Preliminaries section. The highlighted CALL instruction is part of the generated EXEC CICS REWRITE command.

```
CA InterTest for CICS - PROTSYM FILE SOURCE LISTING BREAKPOINT
COMMAND ==>
Program= COBDEMO  Option #          Stmt #          Search=          Margin= 01
-----
----- TASK-PROTCPF                | PROTCPF
----- VSAM-AREA                    |          THIS IS NOT A NAME  PRIME THE
----- REC-LEN                       | +00681.
-----+-----
000916*      LENGTH(REC-LEN)
000917*      END-EXEC.
R   ==>      Call 'DFHEI1' using by content x'0606e0000700004000f0f0f5f2f4
- 000919-    '404040' by reference TASK-PROTCPF by reference VSAM-AREA by
- 000920-    reference REC-LEN end-call.
000921
000922
U 000923 AFTER-REWRITE.
- 000924     MOVE 'THIS IS AFTER REWRITE' TO TASK-TEXT
- 000925     OF TASK-STRUCTURE.
- 000926     GO TO SEND-MAP00.
- 000927 SET-UP-READ.
- 000928     MOVE 681 TO REC-LEN.
```

```
000929*EXEC CICS HANDLE CONDITION NOTOPEN(OPENFIL) END-EXEC.
_ 000930      Call 'DFHEI1' using by content x'0204800007130000000000000000
```

Setting request breakpoints lets you inspect the values of program variables and set options prior to all or specified CICS commands and macros and other program calls.

1. Scroll backward to look at the EXEC CICS REWRITE command.
2. Press PF7.

CA InterTest for CICS displays the following screen.

```
CA InterTest for CICS - PROTSYM FILE SOURCE LISTING BREAKPOINT
COMMAND ==>>
Program= COBDEMO  Option #      Stmt #                      Margin= 01
                        Search=
-----
----- TASK-PROTCPF          | PROTCPF
----- VSAM-AREA              |          THIS IS NOT A NAME PRIME THE
----- REC-LEN                | +00681.
-----
+-----
- 000902      GO TO SEND-REWRITE-RETURN.
- 000903 NUP-READ.
- 000904      PERFORM SET-UP-READ.
- 000905*EXEC CICS READ DATASET(TASK-PROTCPF) INTO(VSAM-AREA) EQUAL
000906*      LENGTH(REC-LEN) RIDFLD(REC-RBA) UPDATE
000907*      END-EXEC.
- 000908      Call 'DFHEI1' using by content x'0602f0000700008400f0f0f5f1f9
- 000909-      '404040' by reference TASK-PROTCPF by reference VSAM-AREA by
000910      reference REC-LEN by reference REC-RBA end-call.
- 000911 VSAM-REWRITE.
- 000912      MOVE 'THIS IS NOT A NAME ' TO VSAM-NAME.
- 000913*EXEC CICS REWRITE
000914*      DATASET(TASK-PROTCPF)
000915*      FROM(VSAM-AREA)
000916*      LENGTH(REC-
LEN)
```

According to the EXEC CICS REWRITE command, the demo program will update the file PROTCPF (specified in the DATASET parameter of the EXEC CICS REWRITE command).



Note: If you do not have the AutoKeep Display facility running, to view the contents of DATASET(TASK-PROTCPF), you can enter Search=TASK-PROTCPF to position the listing where it is defined, and add it to the Keep window (type K next to the line defining TASK-PROTCPF, position the cursor in the field name, and press Enter).

However, you are going to set the No File Updating option to prevent the demo program from updating the file.

To do so, access the Special Options menu.

Set the No File Updating Option

1. As we did when we set and removed the MXR and MXS options, bring up the Program Monitoring screen by specifying the SO option on the command line.
CA InterTest for CICS displays the Special Options menu.
2. Tab to the NUP field.

3. Type an X, and press Enter.

```

CA InterTest MONITORING COMMAND BUILDER - SPECIAL OPTIONS                22
      SET one or more options to override the default monitoring rules in:
      PROG=COBDEMO

      Enter X next to each option desired:

      Source Listing Breakpoint      (SLB) _
      No file updating                (NUP) x
      Reentrancy check                (RNT) _

      Follow monitoring (ON, name, NOPPT) (FOL) -----
      Number of times to be monitored (MUS) -----
      Limit total size of CICS storage (MXS) -----
      Limit total number of CICS requests (MXR) -----

      Set local automatic breakpoint ('*', TERMID, .ANY, OFF) -----
      Limit monitoring to your TERMINAL - '*' or TERMID: -----
      User ID (or .ANY) who will execute the program: -----

      PF1 Help      2          3 End      4 Return   5          6
      PF7           8          9         10       11         12
    
```

CA InterTest for CICS sets this option for the demo program.

4. Press PF4 until you return to the Source Listing Breakpoint screen. Before resuming program execution, look at the MOVE instruction just before the EXEC CICS REWRITE command. It specifies that the data string THIS IS NOT A NAME be moved to the data field VSAM-NAME. If the demo program actually updates the file, that data string displays at the beginning of the first record in that file.
5. Press PF5 to continue program execution. The demo program resumes execution.

Use FILE to Inspect the Record

CA InterTest for CICS now halts the demo program at an unconditional breakpoint you set at procedure name AFTER-REWRITE in the [Demo Preliminaries](#) (see page 142) section.

```

CA InterTest for CICS - PROTSYM FILE UNCOND BEFORE BREAKPOINT
COMMAND ==>
Program= COBDEMO Option #      Stmt #      Search=      Margin= 01
-----+-----
_____ TASK-TEXT OF TASK-STRUCTURE | .. THIS IS A MESSAGE 12/25/99
-----+-----
000922
U 000923 AFTER-REWRITE.
U ==> MOVE 'THIS IS AFTER REWRITE' TO TASK-TEXT
- 000925 OF TASK-STRUCTURE.
- 000926 GO TO SEND-MAP00.
- 000927 SET-UP-READ.
- 000928 MOVE 681 TO REC-LEN.
- 000929*EXEC CICS HANDLE CONDITION NOTOPEN(OPENFIL) END-EXEC.
- 000930 Call 'DFHEI1' using by content x'0204800007130000000000000000
- 000931- '0000000000000000f0f0f5f3f5404040' end-call
- 000932 Service label
- 000933 Go to OPENFIL depending on dfheigdi.
- 000934
- 000935 OPENFIL.
- 000936 IF TASK-SWITCH3 = 'B' GO TO SEND-MAP00.
    
```

```

_ 000937 MOVE 'B' TO TASK-SWITCH3.
_ 000938 MOVE 'PROTCPF NOT OPEN' TO ERROR0.

```

Because a procedure name does not have executable code, the unconditional breakpoint halts execution before the first verb -- in this case, the highlighted MOVE command.

You set the breakpoint at the paragraph immediately following the REWRITE command so you could use the CA InterTest for CICS FILE facility to confirm that the demo program did not, in fact, update the file.

1. Press PF6.
CA InterTest for CICS displays the Breakpoint Primary Option Menu.
2. Select option 1 Main Menu.
CA InterTestv displays the Primary Option Menu.
3. Select 4 Auxiliary storage.
CA InterTest for CICS displays the Auxiliary Storage Menu.

```

----- CA InterTest for CICS AUXILIARY STORAGE MENU -----
OPTION ==> 1

Select an auxiliary storage type, specifying optional criteria below.

1 Files           - Display/select files for access
2 DB2 database    - Invoke DB2 SQL interface facility
3 DL/I database   - Access DL/I database
4 TD queues       - Display/select transient data queues for access
5 TS queues       - Display/select temporary storage queues for access

Type specific or generic file/queue name(s):
(Valid mask characters are * and/or +)

protcpf_-----
.
.

```

4. Type 1 in the Option field and PROTCPF for the file name.
5. Press Enter.
CA InterTest for CICS displays the File Selection menu.
6. Type S next to the file name and press Enter.
CA InterTest for CICS displays the initial File Facility screen.

```

DATATYPE= FC FILEID= PROTCPF MODE= LOG=ON TODEST= PASSWORD=
FUNC= SUBFUNC= RETMETH= ARGTY= SRCHTYP=
MESSAGE=
RETNRCID= CHGLEN=
RCID=
DATA= SIZE= 0000
FORMAT= D 00112233 44556677 8899AABB CCDDEEFF *0123456789ABCDEF*
LOC 0000 .....

```

```

1 Help      2 Format C  3 End      4 BEGB      5          6 DataType DL
7 Page bwd  8 Page fwd  9 Caps Off 10 Top      11 Bottom   12
    
```

7. Press PF4.

CA InterTest for CICS displays the first record in the PROTCPF file.

```

DATATYPE= FC FILEID= PROTCPF MODE=BROWSELOG=ON TODEST=          PASSWORD=
FUNC= NEXT SUBFUNC=          RETMETH=          ARGTY=          SRCHTY=
MESSAGE= CAIN0601 RECORD OBTAINED FOR VIEWING
RETNRCID=40404040404040404040          CHGLEN=
RCID=          SIZE= 02A9
DATA=
FORMAT= D 00112233 44556677 8899AABB CCDDEEFF *0123456789ABCDEF*
LOC 0000 .....
0000 40404040 40404040 4040405C 5C5C4040 ..... *** DSORG=VSKS
0010 404040D9 C5C3D6D9 C440E3D6 40D7D9C9 ..... RECORD TO PRI RECFM=FB
0020 D4C540E3 C8C540C3 C8C5C3D2 D7D6C9D5 ..... ME THE CHECKPOIN LRECL=02A9
0030 E340C6C9 D3C54040 405C5C5C 5C404040 ..... T FILE **** BLKSIZE=0000
0040 40404040 40404040 F0F3F1F4 F0F0F0F0 ..... 03140000 KEYPOS=0000
0050 00000000 00000000 00000000 00000000 ..... KEYLEN=09
0060 00000000 00000000 00000000 00000000 ..... STRNO=02
0070 00000000 00000000 00000000 00000000 .....
0080 00000000 00000000 00000000 00000000 ..... READ
0090 00000000 00000000 00000000 00000000 ..... ADD
00A0 00000000 00000000 00000000 00000000 ..... UPDATE
00B0 00000000 00000000 00000000 00000000 ..... BROWSE
00C0 00000000 00000000 00000000 00000000 ..... DELETE
    
```

```

1 Help      2 Format C  3 End      4 ENDB      5 PREV      6 DataType DL
7 Page bwd  8 Page fwd  9 Caps Off 10 Top      11 Bottom   12
    
```

The previous screen displays the contents of the first record in both hexadecimal and character formats. It is clear from looking at the character display that the data string THIS IS NOT A NAME has not been moved to the beginning of this record

 **Note:** Setting the No File Updating option prevented the demo program from updating this file, without affecting the demo program's source code.

Before returning to the demo program, we will discuss some capabilities of FILE.

FILE Facility

The FILE facility lets you view, update, add and delete records and search for character strings. You can perform these functions at any time (for example, while your program is at a breakpoint). You can also use FILE when no program is executing to perform routine file maintenance.

FILE displays records in dump format, as shown in the previous screen, character format, vertical format, and structured format. To display the record in different formats, press PF2. For structured format, which displays records or DL/I segments on a field-by-field basis, you also must identify the program containing the structure.

You can use FILE with VSAM and BDAM files, DL/I and DB2 databases, temporary storage records, and transient data records.

Take advantage of the FILE facility when testing your own programs. Use FILE to change your test records or to create additional records in the middle of a test session. You can also use FILE to make sure your program has successfully updated a file.

Now we will return to testing the demo program.

Demo Program Completes Execution

1. Press Clear.
CA InterTest for CICS redisplay the File Selection screen.
2. Press PF4 until you return to the Source Listing Breakpoint screen.
3. Press PF5 to resume program execution.
You return to the demo program's Options Menu.

Review What Happened

In this part of the demo session you took advantage of several CA InterTest for CICS features. You were able to do the following:

- Use a request breakpoint to halt the demo program prior to the first File Control REWRITE command.
Request breakpoints make it easy for you to halt your program at various points, such as before all HANDLE CONDITION commands or all Terminal Control commands. When your program is halted, you can inspect main storage or auxiliary storage to detect errors and review program logic.
- Use the No File Updating option to prevent a program from updating a file.
Preventing a program from updating a file means you can test a program repeatedly without having to recreate test data. Many programs can share the same test file because no program will actually change it. And, a test program can even use a production file without corrupting it.
- Display a record in a file.
Displaying a record in a file lets you see whether your program is working correctly. The FILE facility also lets you add, delete, and update records to meet your individual testing needs and maintain files without special one-time programs.

This concludes Option 03 of the advanced demo session. Press Enter to continue with the demo session or press Clear to terminate the session.

Option 04 How to Display Variable Length Data

This section of the demo session details how you can see the current length of variable length records.



Note: Before you begin this section, complete the steps outlined in [Demo Preliminaries \(see page \)](#) (unless you did so earlier in this session). The CA InterTest Demo Session Options Menu should be displayed.

1. Select option 04, and press Enter.
The demo program displays a screen that describes what occurs in this part of the demo session.

```
*****
****
****          CA InterTest Demo Session          ****
****          Display Variable Length Data        ****
****
*****
```

This feature lets you see the current length and contents of variable length data and records at any point in your program. Here is what will happen:

1. CA InterTest halts COBDEMO at an unconditional breakpoint.
2. You use the CORE facility to see the original record.
3. COBDEMO continues to execute and changes the length of the record.
4. You use CORE again to see the updated record.

When this section of the demo completes, you will return to the Options Menu.

Press ENTER to continue or CLEAR to terminate.

2. Press Enter.

CA InterTest for CICS halts the execution of the demo program at an unconditional breakpoint set during the Demo Preliminaries section.

```
CA InterTest for CICS - PROTSYM FILE UNCOND BEFORE BREAKPOINT
COMMAND ==>
Program= COBDEMO Option # Stmt # Margin= 01
Search=
-----
----- VAR-REC-LEN | +00100.
-----+-----
000687**** THIS SECTION OF THE DEMO SHOWS HOW YOU ****
000688**** CAN LOOK AT VARIABLE LENGTH DATA *****
U ==> MOVE +48 TO VAR-REC-LEN.
- 000690 GO TO SEND-MAP00.
- 000691 REPLACE-FILE.
- 000692 IF TASK-SWITCH3 EQUAL SPACE
- 000693 MOVE 'A' TO TASK-SWITCH3
- 000694 MOVE 'DMAP04' TO MAPNAME
- 000695 GO TO SEND-REWRITE-RETURN.
- 000696 IF TASK-SWITCH3 EQUAL 'A'
- 000697 GO TO READ-DATASET
- 000698 ELSE
- 000699 GO TO SEND-MAP00.
- 000700 READ-DATASET.
- 000701*EXEC CICS HANDLE CONDITION
000702* DSIDERR
000703* NOTOPEN(NOT-OPEN)
```

The unconditional breakpoint was actually set at procedure name DO-READ-VAR. Because a procedure name does not have executable code, the unconditional breakpoint halts execution before the first verb in the paragraph. In this case, the first verb is the highlighted MOVE command below the comment lines. Halting execution at this point in the program lets you see the contents of a variable length record before the demo program changes its length. Execution of the highlighted MOVE statement changes the length of the 01-level record VARIABLE-LENGTH-RECORD. Its definition tells us why.

3. Type L VARIABLE-LENGTH-RECORD into the command line, and press Enter.

```
CA InterTest for CICS - PROTSYM FILE UNCOND BEFORE BREAKPOINT
COMMAND ==>
Program= COBDEMO Option # Stmt # Margin= 01
```

```

                                Search=
-----+-----
----- VAR-REC-LEN                | +00100.
-----+-----
000687**** THIS SECTION OF THE DEMO SHOWS HOW YOU ****
000688**** CAN LOOK AT VARIABLE LENGTH DATA *****
U   ==>      MOVE +48 TO VAR-REC-LEN.
-   000690      GO TO SEND-MAP00.
-   000691 REPLACE-FILE.
.
.
.

```

CA InterTest for CICS finds the definition of VARIABLE-LENGTH-RECORD in Working-Storage and displays that portion of the compile listing.

```

CA InterTest for CICS - PROTSYM FILE UNCOND BEFORE BREAKPOINT
COMMAND ==>
Program= COBDEMO  Option #      Stmt #                      Margin= 01
                                Search=
-----+-----
----- VAR-REC-LEN                | +00100.
-----+-----
-   000097                                OCCURS 5 TIMES
-   000098                                INDEXED BY DISTRICT-X.
-   000099 01 VARIABLE-LENGTH-RECORD.
-   000100 03 VAR-REC-LEN                PIC S9(4) COMP.
-   000101 03 VAR-LENGTH-DATA           PIC X
-   000102                                OCCURS 1 TO 100 TIMES
-   000103                                DEPENDING ON VAR-REC-LEN.
-   000104      COPY DFHAID.

```

The definition of VARIABLE-LENGTH-RECORD shows two fields: VAR-REC-LEN and VAR-LENGTH-DATA. Note that the length of VAR-LENGTH-DATA depends on the value of VAR-REC-LEN.

Now we will look at the current contents of VARIABLE-LENGTH-RECORD. You can do this directly from the Working-Storage listing.

4. Tab to the line containing VARIABLE-LENGTH-RECORD.

5. Type D (for display), and press Enter.

```

CA InterTest for CICS - PROTSYM FILE UNCOND BEFORE BREAKPOINT
COMMAND ==>
Program= COBDEMO  Option #      Stmt #                      Margin= 01
                                Search=
-----+-----
----- VAR-REC-LEN                | +00100.
-----+-----
-   000097                                OCCURS 5 TIMES
-   000098                                INDEXED BY DISTRICT-X.
d 000099 01 VARIABLE-LENGTH-RECORD.
-   000100 03 VAR-REC-LEN                PIC S9(4) COMP.
-   000101 03 VAR-LENGTH-DATA           PIC X
-   000102                                OCCURS 1 TO 100 TIMES
-   000103                                DEPENDING ON VAR-REC-LEN.
-   000104      COPY DFHAID.
.
.
.

```

CA InterTest for CICS displays the contents of VARIABLE-LENGTH-RECORD in a structured format.

CA InterTest for CICS - MAIN STORAGE UTILITY - Termid = U031

```
Starting at Address =2080A6F0      Structure Display Format
01 VARIABLE-LENGTH-RECORD        |                               |
02 VAR-REC-LEN                    | +00100.                       |
02 VAR-LENGTH-DATA                | +                               |
```

```
-----
PF1 Help      2          3 End      4 Return    5          6 Dump
PF7 Backward  8 Forward  9 Caps Off 10         11 Redisplay 12 Structure
CORE='000099:VARIABLE-LENGTH-RECORD'
CAIN0467 AT LAST ENTRY IN STRUCTURE
```

As you can see, the structured format gives only one occurrence of each field; it does not tell you the record length. To see the length of a 01-level record, switch from the structured format to the dump format.

6. Press Enter.

CA InterTest for CICS displays the contents of VARIABLE-LENGTH-RECORD in a hexadecimal dump format.

```
CA InterTest for CICS - MAIN STORAGE UTILITY - Termid = U032
Offset      Address      Task
+ 0 00644E4E 4E4E4E4E 4E4E4E4E 4E4E4E4E * ..+++++ * 2030A6F0 184
+ 10 4E4E4E4E 4E4E4E4E 4E4E4E4E 4E4E4E4E * +++++ * 2030A700
+ 20 4E4E4E4E 4E4E4E4E 4E4E4E4E 4E4E4E4E * +++++ * 2030A710 Page
+ 30 4E4E4E4E 4E4E4E4E 4E4E4E4E 4E4E4E4E * +++++ * 2030A720 +000
+ 40 4E4E4E4E 4E4E4E4E 4E4E4E4E 4E4E4E4E * +++++ * 2030A730
+ 50 4E4E4E4E 4E4E4E4E 4E4E4E4E 4E4E4E4E * +++++ * 2030A740
+ 60 4E4E4E4E 4E4E      * +++++ * 2030A750
```

```
-----
PF1 Help      2          3 End      4 Return    5          6 Dump
PF7 Backward  8 Forward  9 Caps Off 10         11         12 SDF
CORE='000099:VARIABLE-LENGTH-RECORD'
```

The dump format shows all occurrences of lower level fields and thus the 01-level record length. This display shows that the record length is 64 hexadecimal (or 100 decimal) bytes. The first two bytes are the field VAR-REC-LEN, and the rest of the bytes are repeated occurrences of the field VAR-LENGTH-DATA.

Now return to the breakpoint to continue execution

7. Press Clear to return to Source Listing Breakpoint screen.

Now you will execute one verb and then stop. This lets you see the contents of the record after the demo program changes its length.

8. Press PF10.

The demo program executes one verb (the MOVE statement, which moves 48 to VAR-REC-LEN). Then CA InterTest for CICS immediately stops the program and displays another breakpoint.

Now the program is halted at the next statement, as shown next.

```
CA InterTest for CICS - PROTSYM FILE STEP BEFORE BREAKPOINT
COMMAND ==>
Program= COBDEMO Option #      Stmt #      Margin= 01
                          Search=
```

```
-----
000688**** CAN LOOK AT VARIABLE LENGTH DATA *****
_ 000689 MOVE +48 TO VAR-REC-LEN.
_ ==> GO TO SEND-MAP00.
_ 000691 REPLACE-FILE.
_ 000692 IF TASK-SWITCH3 EQUAL SPACE
.
.
```

9. To display the contents of VARIABLE-LENGTH-RECORD from the Source Listing screen, you would have to find a line containing the field (such as the definition), type D, and press Enter. This is what you did the first time. Another method is to use a CORE menu:

10. Press PF6.

CA InterTest for CICS displays the Breakpoint Primary Option Menu.

11. Select option 1 Main Menu.

CA InterTest for CICS displays the Primary Option Menu.

12. From the Primary Option Menu, select option 3 Main Storage.

CA InterTest for CICS displays the Main Storage Menu.

13. Select option 3 Breakpoint areas.

CA InterTest for CICS displays a CORE menu for breakpoint-related data.

CA InterTest CORE COMMAND BUILDER - BREAKPOINT-RELATED AREAS (CORE=Bkpt)

Specify area to be displayed, changed or moved: Task number: 00094

COBOL name: variable-length-record_____

Special element: _____	(Enter highlighted keyword)	
SSCR Saved screen image	BMSG Breakpoint message	CMAR Commarea
FCAR Facility cntl area	EIB Exec interface block	TUAR TCT user area
BLLS COBOL BLL cells	CWK COBOL working storage	TGT COBOL TGT
LCL COBOL local storage	DSA COBOL DSA	

SCAN VALUE: _____
 SCAN RANGE: _____ B to scan backwards: _

To VERIFY and/or CHANGE Data:

Existing data: _____
 New data: _____

DATA FORMATS

```
-----
| P' ' = Packed |
| X' ' = Hex     |
| C' ' = Char   |
-----
```

MOVE From _____ To _____

PF1 Help	2	3 End	4 Return	5	6
PF7	8	9 Complex	10	11	12

14. Type VARIABLE-LENGTH-RECORD in the COBOL name field, and press Enter.

CA InterTest for CICS displays the modified contents of VARIABLE-LENGTH- RECORD in structured format.

CA InterTest for CICS - MAIN STORAGE UTILITY - Termid = U031

Starting at Address =2080A6F0	Structure Display Format
01 VARIABLE-LENGTH-RECORD	
02 VAR-REC-LEN	+00048.
02 VAR-LENGTH-DATA	+

.
 .
 .

```
-----
PF1 Help      2          3 End      4 Return    5          6 Dump
PF7 Backward  8 Forward   9 Caps Off 10         11 Redisplay 12 Structure
CORE(00050)='VARIABLE-LENGTH-RECORD'
CAIN0467 AT LAST ENTRY IN STRUCTURE
```

15. Press Enter to display the record in dump format.
The record is now only 30 hexadecimal (or 48 decimal) bytes long because the variable field was reduced from a length of decimal 100 to decimal 48 when the program moved 48 to VAR-REC-LEN.
16. Press Clear.
CA InterTest for CICS displays the Main Storage Menu.
17. Press PF4 until the Source Listing Breakpoint screen displays.
18. Press PF5.
The demo program resumes execution and displays the Options Menu.

Review What Happened

In this part of the demo session you displayed the length and contents of a variable length record.

- You used an unconditional breakpoint to halt program execution before the demo program changed the length of a variable length record.
- You inspected the contents of the record.
- You then executed one verb, halted program execution, and again inspected the contents of the record. This enabled you to confirm its new length.

This concludes Option 04 of the advanced demo session. Press Enter to continue with the demo session or press Clear to terminate the session.

Option 05 How to 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.



Note: Before you begin this section, complete the steps outlined in [Demo Preliminaries](#) (see page 142) (unless you did so earlier in this session). The CA InterTest Demo Session Options Menu should be displayed.

1. Select option 05, and press Enter.
The demo program displays a screen that describes what occurs in this part of the demo session.

```
*****
****
****                      CA InterTest Demo Session                      ****
****                      Work with Table Items                          ****
****
```

CA InterTest makes it easy to display values for indexed, subscripted and qualified data areas. A table defined in WORKING-STORAGE has not been fully initialized. Here is what will happen:

1. CA InterTest halts the program at an automatic breakpoint when it detects an uninitialized table item.
2. You display the table item and correct the contents on the CORE display. You also display the contents of the indexes to determine which table entry caused the error.
3. COBDEMO successfully completes execution.

When this section of the demo continues, you will return to the Options Menu.

Press ENTER to continue or CLEAR to terminate.

2. Press Enter.

The demo program resumes execution.

The next screen displayed is an automatic breakpoint screen. CA InterTest for CICS halted the demo program because it detected a data exception in the highlighted ADD instruction.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>>
Program= COBDEMO  Option #          Stmt #          Search=          Margin= 01
-----
----- DISTRICT-POP(STATE-X COUNTY-X D | ?000.
----- STATE-X                          | ....
----- COUNTY-X                          | ....
----- DISTRICT-X                         | ....
-----+-----
_ 000655      SET COUNTY-X TO 3.
_ 000656      SET DISTRICT-X TO 5.
A ==>>      ADD +1 TO DISTRICT-POP (STATE-X, COUNTY-X, DISTRICT-X).
==>>
==>> ASRA ABEND (0C7) detected and prevented. Caused by invalid decimal
==>> arithmetic data format.
==>>
==>>      Press PF1 for a detailed description.
==>>
_ 000658      MOVE SPACES TO TASK-TEXT OF TASK-STRUCTURE.
_ 000659      MOVE ALL '*' TO TASK-TEXT OF TASK-STRUCTURE-2.
_ 000660      GO TO SEND-MAP00.
_ 000661      INITIALIZE-TABLE.
_ 000662      MOVE SUB-1 TO STATE-NUMBER (SUB-1),

```

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 ASRA 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 and correct the value of the field DISTRICT-POP (STATE-X, COUNTY-X, DISTRICT-X) directly from this screen in the AutoKeep display at the top or by entering D and positioning the cursor under the field name within the code.

1. Overtyping the A to the left of the highlighted statement with D (for display).

- Place the cursor under any letter in DISTRICT-POP, and press Enter.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= COBDEMO  Option #      Stmt #      Search=      Margin= 01
-----
----- DISTRICT-POP(STATE-X COUNTY-X D | ?000.
----- STATE-X | ....
----- COUNTY-X | ....
----- DISTRICT-X | ....
-----+-----
_ 000655 SET COUNTY-X TO 3.
_ 000656 SET DISTRICT-X TO 5.
d ==> ADD +1 TO DISTRICT-POP (STATE-X, COUNTY-X, DISTRICT-X).
==>
==> ASRA ABEND (0C7) detected and prevented. Caused by invalid decimal
==> arithmetic data format.
==>
==> Press PF1 for a detailed description.
==>
_ 000658 MOVE SPACES TO TASK-TEXT OF TASK-STRUCTURE.
_ 000659 MOVE ALL '*' TO TASK-TEXT OF TASK-STRUCTURE-2.
_ 000660 GO TO SEND-MAP00.
_ 000661 INITIALIZE-TABLE.
_ 000662 MOVE SUB-1 TO STATE-NUMBER (SUB-1),

```

CA InterTest for CICS displays the current value of DISTRICT-POP in a structured format. The error message on the bottom line in the following screen tells us this field does not contain a valid value, according to its definition. This is the error that triggered the automatic breakpoint.

Correct an Uninitialized Table Item Dynamically

To dynamically correct this error, modify main storage so the field contains a packed decimal zero. This is the hexadecimal value: 000C. Do this by overtyping the value displayed.

- Tab once to go to the hexadecimal display area.
- Move the cursor to the final 0, overwrite it with C, and press Enter.

```

CA InterTest for CICS - MAIN STORAGE UTILITY - Termid = U031
Starting at Address =2080A626      Structure Display Format
 04 DISTRICT-POP | ?000. |
.
.
.
-----
PF1 Help      2      3 End      4 Return      5      6 Dump
PF7 Backward  8 Forward  9 Caps Off 10      11 Redisplay 12 Structure
CORE='DISTRICT-POP(STATE-X, COUNTY-X, DISTRICT-X)'
CAIN0452 FIELD DOES NOT CONTAIN A VALID PACKED DECIMAL (COMP-3) VALUE

```

CA InterTest for CICS dynamically modifies the value in main storage for DISTRICT-POP (STATE-X, COUNTY-X, DISTRICT-X), and displays the new value. The previous error message no longer displays. You could now resume execution from the breakpoint because you dynamically corrected the error. Before you do, first look at the CORE command displayed above the message line:

```

CORE='DISTRICT-POP(STATE-X, COUNTY-X, DISTRICT-X)'

```

CA InterTest for CICS formatted this command when you used the **d** and the cursor to display the table item. Using the **d** and the cursor meant you did not have to type the complex name. Whenever you can, let CA InterTest for CICS do the work for you.

Display Values of Indexes

Although you corrected the error, you do not know exactly which table entry was at fault. To learn this you will need to know the current contents of STATE-X, COUNTY-X and DISTRICT-X and their occurrences in the table.

1. Return to the breakpoint display to request a display of STATE-X.
2. Press PF3.
CA InterTest for CICS displays the previous Source Listing Breakpoint screen. The program code directly above the breakpoint line shows the values of COUNTY-X and DISTRICT-X. You also could page backward to see the value of STATE-X. However, suppose this code was not there. How would you find the values of STATE-X, COUNTY-X and DISTRICT-X? First, we will see how to display the contents of the variable STATE-X.
3. Tab to the AutoKeep line containing STATE-X.
4. Type D.
5. Press Enter.

```

CA InterTest for CICS - PROTSYM FILE  ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= COBDEMO  Option #          Stmt #                               Margin= 01
                               Search=
-----
d----- DISTRICT-POP(STATE-X COUNTY-X D | ?000.
d----- STATE-X                          | ....
----- COUNTY-X                          | ....
----- DISTRICT-X                         | ....
-----+-----
_ 000655      SET COUNTY-X TO 3.
.
.
.

```

CA InterTest for CICS displays the current contents of STATE-X in dump format, as illustrated in the following figure. CA InterTest for CICS displays the values of indexes in dump format rather than structured format because indexes are not defined in a structure.

```

CA InterTest for CICS - MAIN STORAGE UTILITY - Termid = U032
Offset
+ 0 00000000          * ....          * 20309FC0  189

-----
PF1 Help      2          3 End      4 Return    5          6 Dump
PF7 Backward  8 Forward  9 Caps Off 10          11         12 Structure
CORE='STATE-X'
CAIN4727 Data item is an index set to occurrence 1

```

STATE-X has a value of hexadecimal zero.

The command CA InterTest for CICS generated is simply CORE='STATE-X', as shown on the bottom of the previous figure. This confirms the dump display is showing you the main

storage contents of STATE-X, which is what you wanted.

The last line on the screen indicates STATE-X is set to occurrence 1.

To see the contents of the remaining index variables, COUNTY-X and DISTRICT-X, you could return to the breakpoint display and repeat the request using the **d** and the cursor method. However, a quicker method is to modify the CORE command on the bottom of the screen. All you need to do is change the STATE-X in the CORE command to COUNTY-X.

6. Move the cursor to the CORE command.

7. Overtyping 'STATE-X' with 'COUNTY-X', and press Enter.

CA InterTest for CICS displays the contents of COUNTY-X in dump format and informs you that it is set to occurrence 3, as shown in the following screen. (Note the modified command.)

```

CA InterTest for CICS - MAIN STORAGE UTILITY - Termid = U032
Offset
+ 0 0000001A * .... * 20309FC4 189

-----
PF1 Help      2          3 End      4 Return    5          6 Dump
PF7 Backward  8 Forward    9 Caps Off 10         11         12 Structure
CORE='COUNTY-X'
CAIN4727 Data item is an index set to occurrence 3
    
```

8. To see the value of DISTRICT-X, modify the CORE command to specify DISTRICT-X instead of COUNTY-X.

9. Move the cursor to the CORE command.

10. Overtyping 'COUNTY-X' with 'DISTRICT-X', and press Enter.

CA InterTest for CICS displays the contents of DISTRICT-X and informs you that it is set to occurrence 5. (Note the modified command.)

```

CA InterTest for CICS - MAIN STORAGE UTILITY - Termid = U032
Offset
+ 0 00000008 * .... * 20309FC8 189

-----
PF1 Help      2          3 End      4 Return    5          6 Dump
PF7 Backward  8 Forward    9 Caps Off 10         11         12 Structure
CORE='DISTRICT-X'
CAIN4727 Data item is an index set to occurrence 5
    
```

By displaying the contents of STATE-X, COUNTY-X, and DISTRICT-X, you have determined that DISTRICT-POP (1, 3, 5) is the exact position in the table of the invalid table item.

Now return to the breakpoint display.

11. Press Clear.

The Source Listing Breakpoint screen redisplay.

12. Press PF5.

The demo program resumes execution and returns you to the Options Menu.

Review What Happened

In this part of the demo session the following occurred:

- The demo program was halted at an automatic breakpoint because a table item contained invalid data.
- You corrected the invalid data. However, you did not know the exact position in the table of the invalid table item.
- You used CORE to display the contents of the indexes and their occurrences. This enabled you to pinpoint the position of the invalid table item.

With CA InterTest for CICS, it is easy to debug problems involving indexed, qualified, or subscripted data items because you can display and modify their contents just as you would any other COBOL data item.

This concludes Option 05 of the advanced demo session. Press Enter to continue with the demo session or press Clear to terminate the session.

Option 06 How to Detect a Storage Violation

This section of the test session details how CA InterTest for CICS detects and prevents a storage violation and lets you continue testing.



Note: To perform this section, you should have completed the steps outlined in [Demo Preliminaries \(see page \)](#) . The CA InterTest Demo Session Options Menu should be displayed.

1. Select option 06, and press Enter.

The demo program displays a screen that describes what occurs in this part of the test session.

```

*****
****                                     ****
****                               CA InterTest Demo Session          ****
****                               Storage Violation Detection Option    ****
****                                     ****
*****
    
```

The program attempts to move data from one field to another. However, because the program has relinquished storage for the receiving field, a storage violation would occur. Instead, here is what will happen:

1. CA InterTest halts the program at an automatic breakpoint.
2. You inspect the FREEMAIN instruction which released the storage for the field.
3. You return to the breakpoint display. Then you go around the problem, continuing program execution from the statement following the one that triggered the breakpoint.

When this section of the demo completes, you will return to the Options Menu

Press ENTER to continue or CLEAR to terminate.

2. Press Enter.
CA InterTest for CICS halts the demo program at an automatic breakpoint.

Prevent a Storage Violation

Displayed next is a breakpoint screen. CA InterTest for CICS halted the demo program at an automatic breakpoint. CA InterTest for CICS prevented the storage violation that would have resulted if the highlighted MOVE statement had been executed. CA InterTest for CICS also inserted an A next to the statement to indicate an automatic breakpoint.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>>
Program= COBDEMO  Option #          Stmt #          Margin= 01
                   Search=
-----
----- NEW-DATA          | 00000000 00000000 00000000 00000000
----- STG-AREA1        | 00000000 00000000 00000000 00000000
-----+-----
000968**** NOW MOVE NEW DATA INTO ACQUIRED AREA
000969*
A  ==>  MOVE NEW-DATA TO STG-AREA1.
    ==>
    ==>  an attempt to change an area that does not belong to this task.
    ==>  Possible system damage has been prevented.
    ==>
    ==>  Press PF1 for a detailed description.
    ==>
_ 000971 GO TO SEND-MAP00.
_ 000972 WHICH-ONE.
    
```

The ability of CA InterTest for CICS to prevent storage violations is one of the most important reasons to monitor programs. Storage violations are notorious CICS errors that can do the following:.

1. Bring down your CICS system.
2. Cause a program to produce erroneous data.
3. Cause one program to corrupt the data of another program. In this case, tracking down the problem without CA InterTest for CICS is almost impossible because the error is not in the affected program.

By stopping the program before the storage violation can occur, CA InterTest for CICS protects your system and helps you diagnose and correct the error.

In this example, the statement that would have caused the storage violation is: MOVE NEW-DATA TO STG-AREA1. This statement moves data from NEW-DATA to STG-AREA1. Perhaps the program does not own the area of storage defined as STG-AREA1, so the program cannot move data to that field.

Inspect the FREEMAN Statement

We will page back to find out what caused the error.

1. Press PF7.
CA InterTest for CICS displays the following screen.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>>
Program= COBDEMO  Option #          Stmt #          Margin= 01
                   Search=
    
```

```

----- NEW-DATA | .....V.....
----- STG-AREA1 | .....
-----+-----
000968**** NOW MOVE NEW DATA INTO ACQUIRED AREA
000969*
A ==> MOVE NEW-DATA TO STG-AREA1.
==>
==> an attempt to change an area that does not belong to this task.
==> Possible system damage has been prevented.
==>
==> Press PF1 for a detailed description.
==>
- 000971 GO TO SEND-MAP00.
- 000972 WHICH-ONE.
- 000973 GO TO REPLACE-FILE
- 000974 MXR-OPTION
- 000975 READ-FOR-UPDATE
- 000976 SET-VAR-REC
- 000977 PROCESS-TABLE

```

Look at the FREEMAIN statement that releases the storage for STG-AREA. Now it is clear why the storage violation occurred. The demo program released the storage for STG-AREA. After releasing the storage, the program tried to move data to STG-AREA1, which is part of STG-AREA.

To correct the problem, you would have to remove the FREEMAIN statement or change its location, and then you would recompile the program. However, one of the benefits of CA InterTest for CICS is that you do not have to correct a problem when you discover it. Instead, you can go around the problem and continue testing the program. Now we will redisplay the breakpoint.

2. Press Clear.
CA InterTest for CICS redisplay the automatic breakpoint.

Resume Program Execution

We will continue program execution at the first instruction after the statement that triggered the breakpoint. In the COBOL demo, this is GO TO SEND-MAP00. In the COBOL/II and COBOL/370 demos, this is a GOBACK statement.

1. Tab to the first statement following the breakpoint.
2. Type G (go).
3. Press Enter to continue program execution.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>>
Program= COBDEMO Option # Stmt # Search= Margin= 01
----- NEW-DATA | .....V.....
----- STG-AREA1 | .....
-----+-----
000968**** NOW MOVE NEW DATA INTO ACQUIRED AREA
000969*
A ==> MOVE NEW-DATA TO STG-AREA1.
==>
==> an attempt to change an area that does not belong to this task.
==> Possible system damage has been prevented.
==>
==> Press PF1 for a detailed description.
==>

```

```

g 000971      GO TO SEND-MAP00.
- 000972 WHICH-ONE.
- 000973      GO TO REPLACE-FILE
- 000974      MXR-OPTION
- 000975      READ-FOR-UPDATE
- 000976      SET-VAR-REC
- 000977      PROCESS-TABLE

```

CA InterTest for CICS resumes program execution. The demo program successfully completes execution and displays the Options Menu. By going around the storage violation, you have allowed the demo program to continue executing despite the problem CA InterTest for CICS detected.

Review What Happened

In this part of the demo session the following occurred:

- CA InterTest for CICS detected and prevented a storage violation and halted the demo program at an automatic breakpoint.
- The ability of CA InterTest for CICS to prevent storage violations protects your CICS system and prevents one program from corrupting another.
- By inspecting the statement that would have caused the storage violation, you saw which field was in error. By inspecting the FREEMAIN statement, you were able to determine why the program did not own the storage area.
- Using CA InterTest for CICS to examine the source code surrounding an error makes it easy to learn what caused the problem.
- You went around the problem and continued program execution.
- The ability to go around a problem lets you continue testing and correct many errors in a single test session.

This concludes Option 06 of the advanced demo session. Press Enter to continue with the demo session or press Clear to terminate the session.

Option 07 How to Test a Composite Module

This option details how you can use CA InterTest for CICS to test and debug a composite module. Composite support is an CA InterTest for CICS feature designed to help programmers who are responsible for testing and debugging called subroutines. Skip this option if you are not responsible for these subroutines.

To perform this option, you should have set monitoring for the composite module as described in [Composite Support \(see page 146\)](#) part of Demo Preliminaries.

In this section you will test three programs:

- The main program, COBDEML, the composite module is invoked by the same DEMC transaction as the demo program COBDEMO.

- Two subroutines ASBIN25 and CSBIN25 that are part of composite module COBDEML. However, composite modules at your site could have many called subroutines. You need to tell CA InterTest for CICS only about the ones that you want to test or debug.

Follow these steps:

- In the CA InterTest Demo Session Options Menu, select option 07, and press Enter. The following screen appears:

```

*****
****                                     ****
****                               CA InterTest Demo Session          ****
****                               Composite Support                    ****
****                                     ****
*****

```

Program COBDEML calls two subroutines: ASBIN25 and CSBIN25.

- CA InterTest halts the program at an automatic breakpoint caused by an ASRA. Without composite support, you will see only that you are stopped at the CALL to ASBIN25.
- You abend the task, and then set composite support for COBDEML.
- Type in transaction id DEMC again. This time you will see the actual instruction causing the ASRA within the subroutine.
- You correct the problem that caused the automatic breakpoint.
- You then set an unconditional breakpoint in CSBIN25. When you resume execution, the program is halted at that breakpoint.
- You remove the breakpoint and the program completes execution.

When this section of the demo completes, you will return to the Options Menu.

Press ENTER to continue or CLEAR to terminate.

- Press Enter. The composite module halts at an automatic breakpoint at statement CALL ASBIN25.



Note: If a screen other than the following screen displays, someone at your site has already set composite support for the composite module. In this case, skip to [\(see page \) Re-execute the Demo Program \(see page 180\)](#) .

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING BREAKPOINT
COMMAND ==>
Program= COBDEML  Option #          Stmt #                               Margin= 01
                   Search=
-----+-----
_ 00154
_ 00155 CALL-ASSEMBLER.
A ==>    CALL 'ASBIN25' USING DFHEIBLK.
  ==>
  ==> ASRA ABEND (0C7) detected and prevented. Caused by invalid decimal
  ==> arithmetic data format.
  ==>
  ==>    Press PF1 for a detailed description.
  ==>
_ 00157    MOVE COMM-SW2 TO NUM-CHOICES.
_ 00158
_ 00159 CALL-COBOL.
.
.
.

```

CA InterTest for CICS halted the composite module at an automatic breakpoint because it detected and prevented an error in ASBIN25 when control passed to that subroutine. However, so far you do not know where in ASBIN25 the error occurred.



Note: The name in the Program field is the name of the composite module COBDEML. This is the program CA InterTest for CICS is monitoring -- the main composite program for which you set monitoring before you began the advanced demo session.

Setting composite support lets you test and debug a called subroutine as if it were a separate program, with full symbolic support. That means you can set breakpoints and other monitoring options for that subroutine. When CA InterTest for CICS detects an error, it generates an automatic breakpoint at the statement that triggered the abend, not at the CALL. You can set composite support before you begin to test a program or at any time during testing. In this case, you are going to abend the task, set composite support, and then resume testing.

Abend the Task

When your program is stopped at a breakpoint, you can abend the task rather than resume execution.

1. Type abend in the Command line of the Breakpoint display, and press Enter.
Note: This is the same as selecting option 3 Abend, from the Breakpoint Menu.
CA InterTest for CICS displays the screen shown next.

```

----- CA InterTest for CICS ABEND BREAKPOINTED TASK -----
---
COMMAND ==>

Type an abend code, then press ENTER.

Abend Code ____ Abend code options are:
                blanks      Normal abend, no dump
                XXXX       Abend exits cancelled, no dump
                your code   Your abend code, dump taken
.
.
.
    
```

Because you do not need a dump, you do not need to enter an abend code.

2. Press Enter.
CICS informs you the task has been abended.
3. Press Clear.

Setting Composite Support

To set composite support, access the main CNTL menu from which you can set all of CA InterTest for CICS monitoring options.

1. Type ITST on a clear screen, and press Enter.
CA InterTest for CICS displays the Primary Option Menu.
2. Select option 2 Monitoring.
CA InterTest for CICS displays the Monitoring Menu.
3. Select option 1 Programs.
CA InterTest for CICS displays the Program Monitoring Menu.

```

----- CA InterTest for CICS PROGRAM MONITORING -----
COMMAND ==>

    Type information and S to set or R to remove option(s) below.

    Program . . cobdeml_      Program name (or .ALL, .OPTIONS or generic)
    User ID . . _____    User (or .ANY) for whom the program is monitored

    Option      Description                                          More:  +
    - Status    Display and/or remove monitoring options (S only)
    - Monitor    Monitoring (R removes monitoring and all options previously set)
    - UBP        Unconditional breakpoints (specific program only)
    - CBP        Conditional breakpoints (specific program only)
    - RBP        Breakpoints for CICS, DB2, DL/I or external CALL requests
    - Stmt Trace Statement tracing and data monitoring (COBOL only)
    - New copy   Fetch new copy of program and reset monitoring options (S only)
    - Commands  Indirect commands defined for a specific COBOL or PL/1 program
    - Replace    CICS resource name replacement options
    - Protect    Storage protection monitoring options
    - Special    Other options (storage allocation, file updating, etc.)
    S Composite  Monitor multi-CSECT program's separately compiled components

    PF1 Help    2          3 End      4 Return   5          6
    PF7 Backward 8 Forward  9         10         11         12
    
```

4. In the Program field, type the name of the composite module you are using COBDEML.
5. Type S to the left of Composite.
CA InterTest for CICS displays the Composite Support menu.

```

          CA InterTest for CICS - Composite Support Builder
COMMAND ==>
Define composite support for COBDEML and press PF5.
          SCROLL: PAGE
          Row 00001 of 00003

* *      *      *      *      *      *
- Link name Monitor  Offset  Length  Language  Comments
S COBDEML  COBDEML   20     748    IBMCOB 4.2
S ASBIN25  ASBIN25   768    17C    HLASM 6.0
S CSBIN25  CSBIN25   8E8    674    IBMCOB 4.2

    PF1 Help    2          3 End      4          5 Process  6 RFind
    PF7 Backward 8 Forward  9         10         11         12
    
```

This menu displays link-edit information for the composite program and its subroutines; for example, ASBIN25 and CSBIN25. This information is received from a batch job, executed after the composite module was link-edited.

The batch job that provides the link-edit information identifies all of a program's called subroutines. For efficiency, however, it is a good idea to specify monitor-names just for the subroutines you want to test. You can always set monitoring for additional subroutines by assigning monitor-names on the Composite Support menu.

- Press PF5 to enable composite module monitoring.
CA InterTest for CICS sets composite support for COBDEML and returns you to the Program Monitoring menu.
Now CA InterTest for CICS monitors the subroutines and the composite module so you can take advantage of all CA InterTest for CICS facilities to test and debug each of the programs.

Re-execute the Demo Program

Now you can re-execute the demo program.

- Press PF4.
- CA InterTest for CICS displays the Primary Option Menu.
- Select option X Exit to exit to CICS.
- Clear the screen and type the transaction ID DEMC for the COBOL demo program and press Enter.
CA InterTest for CICS displays the automatic breakpoint at which you were initially halted. However, now the breakpoint occurs at the actual statement in ASBIN25 that triggered the abend, not at the CALL to ASBIN25.

```
CA InterTest for CICS - PROTSYM FILE SOURCE LISTING BREAKPOINT
COMMAND ==>>
Program= ASBIN25  Option #          Stmt #          Displacement=  Margin= 01
                  Search=
-----+-----
      LOC OBJECT CODE  ADDR1 ADDR2  STMT          SOURCE STATEMENT
_ 0000BA
A 0000BA FA20 D1A0 211B 001A0 0011B  ==>          AP   TASKNUM,=P'1'
    ==>
    ==> ASRA ABEND (0C7) detected and prevented. Caused by invalid decimal
    ==> arithmetic data format.
    ==>
    ==>      Press PF1 for a detailed description.
    ==>
_ 0000C0 F920 D1A0 211C 001A0 0011C  282          CP   TASKNUM,=P'2'
.
.
.
```



Note: The name in the Program field is ASBIN25. When control passes to ASBIN25, CA InterTest for CICS starts monitoring that program because you set composite support. Setting composite support enables CA InterTest to identify and display the statement in the subroutine where the error occurred.

The error occurred because of improperly formatted data in TASKNUM. The AP instruction tries to move a packed decimal 1 to TASKNUM. TASKNUM is defined as a packed decimal field, but it contains binary zeros instead of a valid packed decimal value. This is the same type of error you corrected in the basic demo session.

Display the Data in TASKNUM

To correct this error, display the contents of TASKNUM and dynamically change the value.

1. Tab to the AP instruction, and overtype the A with D.
2. Position the cursor under any character in TASKNUM, and press Enter.
CA InterTest for CICS displays the contents of TASKNUM.
On the left of the screen are the Assembler data field names; on the right are the corresponding hexadecimal and character representations of each field.
CA InterTest for CICS displays more than just the contents of TASKNUM. It displays TASKNUM and all the fields below it in the same data structure; that is, in the same DSECT.

Correct the Data in TASKNUM

As you can see, TASKNUM does not contain a packed decimal value. TASKNUM in the COBOL version contains binary zeros rather than a packed decimal value; for COBOL II and COBOL/370 programs, TASKNUM contains other, incorrect values.

You can correct the error by overtyping the incorrect value with a correct value.

1. Tab to the top line and move the cursor to the Hexadecimal value of TASKNUM.
2. Overtyping the current value; set it to a packed decimal zero by typing zero over the "?" and erase the remaining data (erase EOF).
3. Press Enter.

```

CA InterTest for CICS - MAIN STORAGE UTILITY - Termid = U031

Starting at Address = 101DA8  Disp  Structure Display Format
TASKNUM                    1A0  | ?00000.
TSTEXT                     1A3  | .. THIS IS A
                           | MESSAGE
                           | 12/25/99....
                           | .....
                           | .....
                           | .....
                           | .
TSQLEN                     1EC  | +0003866624.
MSGNAME                    1F0  | 0000000000.
MSGLEN                     1F4  | 00000.
RECRBA                     1F6  | .....
RECPONT                    1FC  | 0000000000.

-----
PF1 Help      2      3 End      4 Return    5      6 Dump
PF7 Backward  8 Forward  9 Caps Off 10      11 Redisplay 12 Structure
CORE='R13.TASKNUM'
CAIN4713 R13 not found - have assumed its register value
    
```

CA InterTest for CICS redisplay the screen, changing your entries to uppercase.
Now TASKNUM contains a packed decimal zero. You have corrected the error that triggered the automatic breakpoint.

4. Press Clear.

CA InterTest for CICS redisplay the previous Source Listing Breakpoint screen. However, the explanation of the abend no longer displays.

Set a Breakpoint in Subroutine CSBIN25

Now you are going to set a breakpoint at the first statement in subroutine CSBIN25, the other subroutine you asked CA InterTest for CICS to monitor. Then, when you resume execution, CA InterTest for CICS halts execution at the breakpoint in this subroutine.

When you are stopped in a subroutine, you can use all of the CA InterTest for CICS features, including the following:

- View the program listing and compiler output
- Display and modify main and auxiliary storage
- Set and remove breakpoints and other monitoring options
- Request single-step execution
- Display the backtrace that brought the program to its current point
- Tab back to the Program field.
- Overtyp the current name with the COBOL subroutine name CSBIN25 and press Enter.

```

CA InterTest for CICS - PROTSYM FILE  ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= csbin25  Option #          Stmt #          Displacement=      Margin= 01
                   Search=
-----+-----
      Loc Object Code   Addr1 Addr2  Stmt          Source Statement
      0000E2          2143 001A0 00143  365 CONTINUE DS  0H
      A 0000E2 FA20 D1A0 2143 001A0 00143  ==>      AP   TASKNUM,=P'1'
      .
      .
      .

```

CA InterTest for CICS displays the source listing for COBOL program, for example, CSBIN25. Now you are going to set an unconditional breakpoint at the first statement in the Procedure Division.

- Tab to the MOVE ZERO TO DIVCT statement.
- Type U to the left of that statement, and press Enter. The U sets a breakpoint that stops *before* the statement executes. You could also enter a) to set a breakpoint that stops *after* the statement executes.

```

CA InterTest for CICS - PROTSYM FILE  ABEND DETECTED DISPLAY
COMMAND ==>
Program= CSBIN25  Option #          Stmt #          Margin= 01
                   Search=
-----+-----
      000017      05  COMM-DATE.
      000018          10  COMM-MM          PIC 99.
      000019          10  FILLER          PIC X.
      000020          10  COMM-DD          PIC 99.

```

```

- 000021          10  FILLER          PIC X.
- 000022          10  COMM-YY        PIC 99.
- 000023 PROCEDURE DIVISION USING COMM-TEXT.
u 000024          MOVE ZERO TO DIVCT.
- 000025          GET-DATE.
- 000026          MOVE COMM-YY TO PROGY.
- 000027          MOVE COMM-MM TO PROGMM.
- 000028          MOVE COMM-DD TO PROGDD.
- 000029          DIVIDE PROGY BY 4 GIVING DIVYY REMAINDER YY-REM.
- 000030          MOVE PROGMM TO DIVMM.
- 000031          GOBACK.
An "M" preceding a data-name reference indicates that the data-name is modified

-   DEF DATA NAMES          REFERENCES
-   17 COMM-DATE

```

CA InterTest for CICS sets an unconditional breakpoint at the MOVE statement.

- Press Clear.
CA InterTest for CICS redisplay the previous Source Listing Breakpoint screen.

Resume Program Execution

Now you are going to continue program execution.

Press PF5.

The program resumes execution. When the control passes to subroutine CSBIN25, CA InterTest for CICS halts the program at the unconditional breakpoint you just set.

At this point you could use CA InterTest for CICS to examine the program listing of CSBIN25 or the values of any program variables. However, you are simply going to remove the breakpoint and resume program execution.

Remove the Breakpoint and Resume Execution

Overtyp the U with X, and press PF5.

CA InterTest for CICS removes the unconditional breakpoint and program execution resumes.

COBDEML completes execution and the Options Menu is displayed.

Review What Happened

In this part of the demo session the following occurred:

- When you executed COBDEML, CA InterTest for CICS detected and prevented an abend caused by invalid data in one of its called subroutines.
CA InterTest for CICS displayed the automatic breakpoint at the statement that called the subroutine, rather than the statement that triggered the breakpoint, because CA InterTest for CICS was not monitoring the subroutine.
- You set composite support for COBDEML so that CA InterTest for CICS would monitor its subroutines: ASBIN25 and CSBIN25.
Setting composite support lets you test and debug a called subroutine with full symbolic support and use all CA InterTest for CICS features.
- When you re-executed COBDEML, CA InterTest for CICS redisplayed the automatic breakpoint; this time at the actual statement in subroutine ASBIN25 that caused the error.
- You corrected the error by overtyping the contents of the invalid field.

- You then set an unconditional breakpoint in the other subroutine, CSBIN25, and resumed program execution. When control passed to CSBIN25, CA InterTest for CICS halted execution at the breakpoint you set.
The ability to halt execution at any statement in a subroutine means you can test a subroutine just as if it were a main program.
- You removed the unconditional breakpoint and continued program execution.

This concludes Option 07 of the advanced demo session. Press Enter to continue with the demo session or press Clear to terminate the session.

PL/I Primer

The main purpose of this PL/I Primer is to train new users in the basic product features used to test and debug programs. The primer also introduces some of the advanced features of CA InterTest for CICS.

We recommend that all users take the demo sessions in this article and become familiar with the features explained here. For complete information about these and other features, see [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging) . We suggest that you read the [PL/I Basic Demo Session \(see page 184\)](#) first, because it is the best way to begin learning about your product.

Getting Help

The Help facility is online documentation of product features. It makes it easy to learn and use the product. Help is available from all product screens.

To view an online summary of new features for this release, use ITST Option 8 What's New.

PL/I Basic Demo Session

This article takes you step-by-step through the basic CA InterTest for CICS demo session. Performing the demo at a terminal is the best way to start learning about CA InterTest for CICS.

The basic demo session illustrates many of the testing and debugging tasks you will use on your own programs. These tasks are discussed in more detail in the next article. For more information on any of these topics , see [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging) .

We are assuming that CA InterTest for CICS has been installed on your system. If not, contact your systems programmer.

- [Maintain Synchronized Processing \(see page 185\)](#)
- [Demo Session Objectives \(see page 186\)](#)
- [Demo Session Scenario \(see page 186\)](#)
- [Select the Source Program from the ITST Menus \(see page 187\)](#)
- [View the Source Listing Profile \(see page 190\)](#)

- [Set Monitoring \(see page 190\)](#)
- [View Monitoring Status \(see page 191\)](#)
- [Begin Program Execution \(see page 192\)](#)
- [Detect and Prevent an Abend \(see page 193\)](#)
- [Determine the Cause of the Error \(see page 193\)](#)
- [Change the Value in TASKNUM Dynamically \(see page 194\)](#)
- [Control Program Execution \(see page 196\)](#)
- [Set Unconditional Breakpoints \(see page 197\)](#)
- [What You Learned \(see page 200\)](#)

Maintain Synchronized Processing

Before you begin testing, you should be aware that when CA InterTest for CICS searches the symbolic files for the PL/I programs you specify for testing, it tries to match the date and time in the symbolic file to that of the load module. If a match cannot be found, one of the following is displayed:

- A Symbolic Version List
- A warning message

This feature lets you maintain synchronized processing at all times by enabling you to select the correct symbolic version of the program that you want to test.

Symbolic Version Processing

When you request a PL/I program that has no previously declared breakpoints or monitoring options set, CA InterTest for CICS matches the most recently compiled symbolic file date and time to the load module date and time.

- If an exact match is found, CA InterTest for CICS automatically selects and displays the program. However, if an exact match is found and a more recently compiled version of the program is in a PROTSYM file, CA InterTest displays the symbolic version list from which you can do the following:
 - Select the matching listing and debug the program.
 - Do not select the matching listing and cancel monitoring. Then new copy the latest version of the program into your CICS region and select the latest program version for monitoring.
- If a match is not found, CA InterTest for CICS displays the Symbolic Version List from which you can:
 - Select the appropriate symbolic file.
 - Ignore symbolic processing for the program.

The Symbolic Version List screen also explains the cause of the mismatch.

During Automatic Breakpoint processing, if a program has no previously selected symbolic file, CA InterTest for CICS matches the load module's date and time to a symbolic file.

- If an exact match is found, CA InterTest for CICS automatically selects and displays the program.

- If a match is not found, CA InterTest for CICS automatically selects the first symbolic file that contains the program and displays it with a warning message on the first Source Listing Breakpoint screen indicating the mismatch. This is shown on the following screen.

```

CA InterTest for CICS - SYMBOLIC VERSION LIST

  Program = PL1DEMO      Load Module Date/Time = 11/12/2000 09:02:03

  File ID   Date       Time       Language   Comments
  _ PROTSYM 11/12/2000 09.52.23  PLI        LATEST VERSION
  _ PROTDMO 08/31/2000 14.12.15  PLI
  .
  .
  .
  S - Select which Symbolic file to use
-----
PFKEYS:  1 Help      2           3 No file   4           5           6
          7           8           9           10          11          12
CAIN8000 The latest Symbolic version does not match the current load module

```



Note: Once a symbolic file is selected for a program, CA InterTest for CICS continues to use the selected file and bypasses subsequent date and time matching until all declared breakpoints and monitoring options are removed for the program, or until a NEWCOPY command is executed. This can be done from ITST Option 2.1 by selecting the Option for New copy, or by typing the following command from CICS:

CNTL=NEW, PROG=programe

Demo Session Objectives

This demo session, teaches you how to perform the following tasks:

- Select a source listing from the CA InterTest for CICS ITST menus
- Inform CA InterTest for CICS that you want to test a program (set monitoring)
- Respond to the information CA InterTest for CICS provides when it detects an error
- Examine the value of a variable
- Dynamically change the value of a variable
- Halt program execution at any point (set breakpoints)
- Resume program execution

Demo Session Scenario

The CA InterTest for CICS basic demo session takes you through the following scenario:

1. You use the CA InterTest for CICS Primary Option Menu to select and view the source listing of PL1DEMO, the sample program to execute.

2. You set monitoring for the sample program and view the status display of your monitoring request.
3. You exit CA InterTest for CICS menus and initiate PL1DEMO.
4. CA InterTest for CICS intercepts and prevents an ASRA abend in PL1DEMO. CA InterTest for CICS halts PL1DEMO at an assignment statement and displays a diagnostic message informing you that the problem was caused by improperly formatted data.
5. You examine the current value of TASKNUM, the receiving field in the assignment statement that you suspect might be the cause of the problem. You find that TASKNUM does not contain a valid packed decimal number.
6. You move a packed decimal zero into TASKNUM to correct the error.
7. You set a breakpoint to halt PL1DEMO at another statement. This step is purely instructional; it has nothing to do with correcting the ASRA.
8. You resume program execution. CA InterTest for CICS halts the program at the breakpoint you set in Step 7. You can see that when the assignment statement executed, the value of TASKNUM changed.
9. You clean up by removing the breakpoint you set. You then resume program execution and PL1DEMO runs to completion.

Select the Source Program from the ITST Menus

Start your session by selecting the source listing of the CA InterTest for CICS demonstration program from the ITST Primary Option Menu.

1. Sign on to CICS.
Type ITST on a clear screen.
2. Press Enter.
The Primary Option Menu appears.

```

----- CA InterTest for CICS PRIMARY OPTION MENU -----
OPTION  ==>
  1 Source          - Display/select program source files/listings
  2 Monitoring      - Display/modify CA InterTest monitoring/activity
  3 Main storage    - Display/modify CICS storage areas
  4 Auxiliary storage - Display/access databases/files/queues
  5 Dump analysis   - Invoke CA SymDump CICS dump/trace capture facility
  6 Product help    - Invoke CA InterTest product help facility
  7 Status/Maintenance - Product status and maintenance functions
  8 What's new?     - Display information about CA InterTest for CICS
  X Exit           - Terminate menu processing
    
```

```

PF1 Help      2          3 End        4 Return     5           6
PF7           8          9           10          11          12
    
```

Notice the PF keys have conventional assignments for ISPF-like navigation:

- PF1 Help accesses online help

- PF3 End returns one panel
- PF4 Return returns to the top-level menu

Now we can look at the source listing of the program we will debug.

1. Type 1 in the Option field.
2. Press Enter.
The CA InterTest for CICS Source Menu appears.

```

----- CA InterTest for CICS SOURCE MENU -----
OPTION ==> 1

  Select a member list type, specifying optional criteria below.

  1 Source listings   - Display/select program source listings
  2 Symbolic files   - Display/select program source SYMBOLIC files

  Type specific or generic program/file name(s):
  (Valid mask characters are * and/or +)

  p*-----
  .
  .
  .
    
```

3. Type 1 in the Option field to search for program listings.
4. Tab to the first entry field for file/program names, and type P*. The asterisk is a generic or wildcard character. It indicates that anything from this point on in a file name will satisfy the search criteria.
5. Press Enter.
The Source Listing Selection screen displays. It lists all program Source Listings beginning with the letter P, in all symbolic files. Your list might have fewer or more entries.

```

----- CA InterTest for CICS SOURCE LISTING SELECTION -----
COMMAND ==>

  Type S to select a source listing.

  Name      File      Created      Size Attributes
  _ PL1DEML  PROTSYM  09/06/2013  14:03   43 PL/I 3.2, composite
  _ PL1DEMO  PROTSYM  09/06/2013  14:01  268 PL
  /I 3.2
  _ PROTCI63 PROTSYM  05/25/2013  12:53   637 HLASM 5.0, no purge
  _ PROTCI64 PROTSYM  05/25/2013  12:53   642 HLASM 5.0, no purge
  _ PROTIN62 PROTSYM  05/25/2013  12:53   257 HLASM 5.0, no purge
  _ PROTIN63 PROTSYM  05/25/2013  12:53   258 HLASM 5.0, no purge
  _ PROTIN64 PROTSYM  05/25/2013  12:54   259 HLASM 5.0, no purge
  _ PSBIN25  PROTSYM  09/06/2013  16:15    25 PL/I 3.2
  _ *** End of data ***

  PF1 Help      2 Refresh     3 End         4 Return      5             6
  PF7 Backward  8 Forward     9             10           11           12
    
```

6. Locate the demo program, PL1DEMO, in the Selection List. You might need to scroll the file list using PF7 and PF8. Optionally, you can use the command:

LOCATE program-name



Note: If the file list does not contain the name of the demo program indicated above, it could be because the person who installed CA InterTest for CICS changed the name of the CA InterTest for CICS sample programs available to you.

Check with that person to find out the correct names.

7. Type S in the field to the left of the program name to select it for Source Listing Display.

8. Press Enter.

CA InterTest for CICS displays the PROTSYM File Source Listing Display, which shows the compiled listing of PL1DEMO.

- The top half of the display identifies the program and provides fields for entering commands, option numbers (to jump to different parts of your source listing), statement numbers, and search criteria.

Depending on your session defaults, you might not see lines identifying the PF key and Option # field titles. These were omitted from the images in this article for clarity.

- The bottom half of the display contains the source listing.

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY

COMMAND ==>

Program= PL1DEMO Option # Stmt # Search= Margin= 01

```
-----+-----
- 18.1          /* INTEREST PL/I DEMO PROGRAM 10/01/89 3.1.200
- 19.1          PL1DEMO: PROC(DFHEIPTR,COMMAREA_PTR) OPTIONS(MAIN REENT
- 20.1    1     DCL 1 DFHCNSTS STATIC, /* CONSTANTS USED BY TRANSLATOR
- 21.1          2 DFHLDVER CHAR(22) INIT('LD TABLE DFHEITAB 530.
- 22.1          2 DFHEIB0 FIXED BIN(15) INIT(0),
- 23.1          2 DFHEID0 FIXED DEC(7) INIT(0),
- 24.1          2 DFHEICB CHAR(8) INIT(' ');
- 25.1    1     DCL DFHEPI ENTRY, DFHEIPTR PTR;
- 26.1    1     DCL 1 DFHEIBLK BASED (DFHEIPTR),
- 27.1          02 EIBTIME FIXED DEC(7),
- 28.1          02 EIBDATE FIXED DEC(7),
- 29.1          02 EIBTRNID CHAR(4),
- 30.1          02 EIBTASKN FIXED DEC(7),
- 31.1          02 EIBTRMID CHAR(4),
- 32.1          02 EIBFIL01 FIXED BIN(15),
- 33.1          02 EIBCPOSN FIXED BIN(15),
- 34.1          02 EIBCALEN FIXED BIN(15),
- 35.1          02 EIBAID CHAR(1),
- 36.1          02 EIBFN CHAR
(2),
```



Note: The statement numbers on your screen might not match those shown in the illustrations. This will not hinder you from performing the test session.

View the Source Listing Profile

Type PROFILE on the Command line and press Enter (or, press PF4 Profile) to view the PF key assignments and other session options.

CA InterTest for CICS displays the Source Listing Profile for your current session. This lists the PF keys, options, and current environment settings relating to the Source Listing display.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING PROFILE
COMMAND ===>
Program= PL1DEMO Option #      Stmt #      Search=      Margin= 01
OPTS 1 Data xref 2 Aggregate 3 Stor reqs 4 Stat stor 5 Var stor  More:  +
      6 Offsets  7 Gend code 8 Err msgs  9 Srch fwd 10 Srch bwd
PFKS 1 Help      2          3 End      4 Auto prms 5 Monitor  6 Menu
      7 Backward 8 Forward  9 Next Wnd 10          11          12 Status
-----+-----
Display window = N          N (None), T (Titles), R (Registers),
                  K (Keep), P (Program)
PF7/8 amount   = PAGE      PAGE, HALF, STOP, or a number from 1 to 9999
Step Timing    = BEFORE    Stop Before or After the next STMTS is executed
Stepping amount = 001      The number of STMTS to execute
Auto-stepping  = OFF       ON to activate; press PF4 to change values
Source List BKPT = ON      OFF to use the detailed breakpoint display
From terminal ID = U006    Terminal ID where the program will execute
BKPT terminal ID = U006    Terminal ID to receive the breakpoint displays
User ID        = .ANY     User ID who will execute this program
AutoKeep Display = ON     OFF to deactivate
Code Counting  = OFF     ON to activate Code Coverage
SDF            = DATA    DATA for Structure Display Format
    
```



Note: Verify that the AutoKeep Display feature is activated for this demo session. If not, change it from OFF to ON and press Enter.

Set Monitoring

The next step is to instruct CA InterTest for CICS to monitor PL1DEMO. As part of monitoring a CICS command-level program such as PL1DEMO, CA InterTest for CICS automatically prevents all CICS abends during execution. How this works is detailed later when CA InterTest for CICS prevents PL1DEMO from abending because of an ASRA.

There are a few ways to tell CA InterTest for CICS to monitor the program. If you are currently viewing a program's source listing, you can do the following:

- Type MONITOR on the Command line and press Enter.
- Press PF5 (notice PF5 is listed as Monitor on the Source Listing Profile illustration).

Press PF5 or use the monitor command to begin monitoring the demo program. You can do this from either the Source Listing Profile or Source Listing Display.

The screen momentarily flashes as the request processes. The Source Listing Display screen then reappears.

View Monitoring Status

To verify that the demo program is being monitored, display the Monitoring Status report for the current program.

1. Press PF12, or type STATUS on the command line and press Enter.

```
CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==> status
Program= PL1DEMO Option #      Stmt #                               Margin= 01
                               Search=
-----+-----
_   18.1          /* INTERTEST PL/I DEMO PROGRAM 10/01/89 3.1.200
_   19.1          PL1DEMO: PROC(DFHEIPTR,COMMAREA_PTR) OPTIONS(MAIN REENT
.
.
.
```

CA InterTest for CICS displays the Monitoring Status screen. This status shows the monitoring entry for the current program only.

This display gives you an up-to-date summary of how CA InterTest for CICS is monitoring your program, in an expandable/collapsible tree format. You can scroll the list, collapse or expand it using the indicated PF keys at the bottom of the display.

```
----- CA InterTest for CICS MONITORING STATUS -----
COMMAND ==>
```

Type + to expand or - collapse option levels displayed below,
or R to remove option(s).

Option	Description	Attributes
- PL1DEMO	Program monitor entry	PL/I
- -.ANY	User monitoring options	Active
	Symbolic listing file	PROTSYM
- -SLB	Source listing breakpoints	U006
	*** End of data ***	

```
PF1 Help      2 Refresh    3 End         4 Return     5 Collapse   6 Expand
PF7 Backward  8 Forward    9            10          11          12
```

Each entry in the monitoring status display for PL1DEMO indicates the following information:

- The top entry tells you CA InterTest for CICS is monitoring the PL/I program PL1DEMO.
- The second entry (.ANY) indicates CA InterTest for CICS monitors PL1DEMO when any user executes it. If your status display shows your CICS User ID instead of .ANY, that is fine, because it indicates monitoring and options taking effect whenever your ID is executing the program, but not when anyone else executes the program. In effect, when the user monitoring option is set to a specific CICS User ID, CA InterTest for CICS creates a personal debugging session for that user.



Note: For details on using and switching the User ID Monitoring Option, see [Monitoring Menu Options \(https://docops.ca.com/display/CAITSD11/Monitoring+Menu+Options\)](https://docops.ca.com/display/CAITSD11/Monitoring+Menu+Options).

Detect and Prevent an Abend

The next screen is not displayed by the PL/I demo. It is a diagnostic screen, called a breakpoint display, which is displayed by CA InterTest for CICS when it detects an error.

```

CA InterTest for CICS - PROTSYM FILE  ABEND DETECTED BREAKPOINT
COMMAND ==>>
Program= PL1DEMO  Option #          Stmt #          Margin= 01
                               Search=
-----
----- TASKNUM                | ?00000.
-----+-----
- 636.1          CONTINUE_TASK:
- 637.1          MAPNUM  = '00';
A ==>.1         1          TASKNUM = TASKNUM + 1;      ==>
==> ASRA ABEND (0C7) detected and prevented. Caused by invalid decimal
==> arithmetic data format.
==>
==>          Press PF1 for a detailed description.
==>

- 639.1         1          IF TASKNUM > 2 THEN
- 640.1         1          GOTO END_SESSION;
- 641.1         1          IF TASKNUM = 1 THEN
- 642.1         1          GOTO SENDSCR3;
- 643.1         1          GOTO END_SESSION;
- 644.1
- 645.1         1          SENDSCR3:                      /** ASRA CONGRATULAT
- 646.1          /* EXEC CICS SEND MAP('DASRA'

```

Notice that CA InterTest for CICS has highlighted an ADD instruction and displayed a message below it. Execution of that ADD instruction triggered an ASRA abend. CA InterTest for CICS prevented the abend and then displayed the diagnostic screen you are currently viewing.

When CA InterTest for CICS stops program execution, we say that it halts the program at a breakpoint. It does this automatically, as in the example, or it halts a program at a breakpoint set by you, the programmer. The A to the left of the assignment statement indicates that the current halt in program execution is an Automatic breakpoint - not one set by you.

Now look at the highlighted message. It explains that the problem was caused by improperly formatted data. Which data? Since `TASKNUM = TASKNUM + 1` triggered the breakpoint, it is likely that `TASKNUM` contains improperly formatted data.

So far, CA InterTest for CICS has prevented PL1DEMO from abending and identified the problem. Next, you will perform the following procedures:

1. Determine the cause of the error.
2. Dynamically correct the error by changing the value of `TASKNUM`.

Determine the Cause of the Error

Confirm that the value stored in `TASKNUM` is not in valid arithmetic format by displaying its current value; that is, its value prior to the execution of the assignment statement that triggered the ASRA.

CA InterTest for CICS displays the current value of `TASKNUM`, as shown in the following screen.

```

CA InterTest for CICS - PROTSYM FILE  ABEND DETECTED BREAKPOINT
COMMAND ==>>
Program= PL1DEMO  Option #          Stmt #          Margin= 01
                               Search=

```

```

-----
----- TASKNUM | ?00000.
-----+-----
- 636.1          CONTINUE_TASK:
- 637.1          MAPNUM = '00';
A ==>.1 1        TASKNUM = TASKNUM + 1;    ==>
==> ASRA ABEND (0C7) detected and prevented. Caused by invalid decimal
==> arithmetic data format.
==>
==> Press PF1 for a detailed description.
==>

- 639.1 1        IF TASKNUM > 2 THEN
- 640.1 1          GOTO END_SESSION;
- 641.1 1        IF TASKNUM = 1 THEN
- 642.1 1          GOTO SENDSCR3;
- 643.1 1          GOTO END_SESSION;
- 644.1
- 645.1 1        SENDSCR3:                /** ASRA CONGRATULAT
- 646.1          /* EXEC CICS SEND MAP('DASRA')

```

When the AutoKeep Display feature is active, CA InterTest for CICS automatically displays the value(s) of the variable(s) associated with a line of code. In this case, TASKNUM displays in this specially formatted Keep window. On the left is the name of the data item; on the right the question mark in the first position indicates incorrect data content for the fields' datatype, in this case low values (binary zeros).

Up to six data items at a time display within the scrollable Keep window. This feature lets you see how the values of data items change as your program executes. The Keep window remains until you remove all data items from it.

Now, look at the contents of TASKNUM. It does not contain a valid packed decimal value. Instead, it contains low-values (binary zeros). PL/I does not let you assign a value to a field without initializing it.

Change the Value in TASKNUM Dynamically

Now that we have identified and confirmed the cause of the problem, we can fix it.

1. Type M (modify) to the left of TASKNUM in the Keep window.
2. Press Enter.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= PL1DEMO  Option #      Stmt #      Search=      Margin= 01
-----
m_____ TASKNUM | ?00000.
-----+-----
- 636.1          CONTINUE_TASK:
- 637.1          MAPNUM = '00';
A ==>.1 1        TASKNUM = TASKNUM + 1;    ==>
.
.
.

```

CA InterTest for CICS generates a fill-in-the-blanks MOVE statement, as illustrated in the following figure. The generated MOVE statement is equivalent to the PL/I assignment statement:

TASKNUM = _____

- Type zeros in the MOVE field, as shown next, and press Enter.

 **Note:** You do not have to know the type of data (binary, packed, and so on) or the length of TASKNUM. The CA InterTest for CICS MOVE statement automatically takes care of that for you.

```
CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>>
Program= PL1DEMO  Option #      Stmt #                      Margin= 01
                   Search=
-----
----- TASKNUM                      | ?00000.
-----+-----
```

```
MOVE zeros-----
to
TASKNUM
```

Overtyping Underscores with a Data-Name, Figurative Constant, Alphanumeric Literal, or Numeric Literal

CA InterTest for CICS executes the MOVE statement. As a result, TASKNUM now contains a packed decimal zero. CA InterTest for CICS also displays a main storage display that shows the new value of TASKNUM, as shown in the following screen.

CA InterTest for CICS - MAIN STORAGE UTILITY - Termid = U006

```
Starting at Address =20804231      Structure Display Format
02 TASKNUM                        | 00000. |
02 TS_TEXT                         | ..... |
                                   | ..... |
                                   | ..... |
```

```
-----
PF1 Help      2          3 End      4 Return    5          6 Dump
PF7 Backward  8 Forward  9 Caps Off 10          11 Redisplay 12 Structure
CORE=MOVE ZEROS TO TASKNUM
CAIN0201 RECEIVING FIELD has been changed as shown
```

You can change the contents of a field by overtyping the desired bytes in the Keep window or on the main storage display. Overtyping is faster than using a CA InterTest for CICS-generated MOVE statement. However, with the MOVE statement you do not have to know the internal representation of the data. CA InterTest for CICS automatically takes care of the details for you.

Now we return to the program listing so that we can continue to test PL1DEMO.

- Press Clear. CA InterTest for CICS redisplay the breakpoint screen without the explanation of the abend. Note the new value of TASKNUM in the Keep window.





Note: You also can correct this bug using indirect commands, which are described in [PL/I Advanced Monitoring Features \(see page 220\)](#).

Control Program Execution

Now that the value in TASKNUM has been properly initialized, the next step might be to continue testing by resuming program execution. However, this is a good opportunity to learn about another important feature of CA InterTest for CICS -- the ability to control program execution by setting breakpoints.

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 initiate the task, and the program either runs to completion or abends.

With CA InterTest for CICS, you can control program execution in a number of ways. For example, you can set stops, called breakpoints, anywhere in your program. The types of breakpoints you can set for PL/I programs are unconditional, conditional, and request breakpoints.

- When you set an *unconditional* breakpoint at a statement, the program stops just before or just after the statement is executed. (Depending on the type of unconditional breakpoint that you set.)
- When you set a *conditional* breakpoint at a statement, the program stops only if a condition you specified is met, such as a counter equaling or exceeding some value. You can also set a conditional breakpoint to stop at any instruction when the condition you specified is met.
- When you set a *request breakpoint*, the program stops before all CICS commands, macros, and other program calls (such as calls to DL/I or DB2), or just before specific commands (such as all READ or WRITE commands).

What You Can Do When Your Program Is Stopped

Once a program is stopped, you can use the CA InterTest for CICS testing and debugging facilities to do the following tasks:

- Examine the source listing
- Examine and modify main and auxiliary storage to detect and correct errors
- Set and remove breakpoints
- Examine a program's backtrace, or execution path
- Keep variables in a Keep window to observe changes in their values
- Abend your task with or without a dump
- Go around a problem by resuming program 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 one verb and then stops

Now we will demonstrate how easy it is to control program execution by setting an unconditional breakpoint.

Set Unconditional Breakpoints

You can set unconditional breakpoints directly on the Source Listing screen just as easily as you displayed and modified TASKNUM. Enter U or) to the left of each statement where you want the application to halt program execution. The U sets a breakpoint that stops *before* the statement is executed. The) sets a breakpoint that stops *after* the statement is executed.

Let us see how to set a "before" breakpoint at the following IF statement:

IF TASKNUM = 1 THEN

1. Type **U** to the left of the statement number on the IF TASKNUM = 1 THEN line.
2. Press Enter.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>>
Program= PL1DEMO Option # Stmt # Search= Margin= 01
-----
----- TASKNUM | 00000.
-----+-----
- 636.1 CONTINUE_TASK:
- 637.1 MAPNUM = '00';
A ==>.1 1 TASKNUM = TASKNUM + 1;
- 639.1 1 IF TASKNUM > 2 THEN
- 640.1 1 GOTO END_SESSION;
u 641.1 1 IF TASKNUM = 1 THEN
- 642.1 1 GOTO SENDSCR3;
- 643.1 1 GOTO END_SESSION;
- 644.1
- 645.1 1 SENDSCR3: /** ASRA CONGRATULAT
- 646.1 /* EXEC CICS SEND MAP('DASRA')
- 647.1 MAPSET('IN25PMP')
- 648.1 WAIT
- 649.1 ERASE */
- 650.1 DO;
- 651.1 1 1 DCL DFHENTRY_BF5EF358_068E978B BASED(ADDR(DFHEI0)
- 652.1 ASSEMBLER) ENTRY(*,CHAR(7),*,FIXED BIN(15),CHAR(7));

```

3. Press PF5 (Resume) to continue execution from where the program is currently stopped (at the statement TASKNUM = TASKNUM + 1).

CA InterTest for CICS resumes program execution and continues until it reaches the breakpoint you just set. CA InterTest for CICS halts the program before that statement executes and displays the following screen.

```

CA InterTest for CICS - PROTSYM FILE UNCOND BEFORE BREAKPOINT
COMMAND ==>>
Program= PL1DEMO Option # Stmt # Search= Margin= 01
-----
----- TASKNUM | +00001.
-----+-----
- 639.1 1 IF TASKNUM > 2 THEN
- 640.1 1 GOTO END_SESSION;
U ==>.1 1 IF TASKNUM = 1 THEN
- 642.1 1 GOTO SENDSCR3;
- 643.1 1 GOTO END_SESSION;

```

.
.

Notice the **U** to the left of the highlighted statement. It identifies the breakpoint as an unconditional "before" breakpoint. For an "after" breakpoint, the application displays a).



Note: The value of TASKNUM is now 1 because the TASKNUM = TASKNUM + 1 statement executed successfully.

When you are stopped at a breakpoint, you have many choices. You can do the following tasks:

- Scroll or search through your source listing
- Examine and modify main and auxiliary storage
- Add a variable to the Keep window using the k line command
- Set and remove breakpoints
- Examine the program's Backtrace Summary
- Abend your task (with or without a dump)
- Go around a problem by resuming program execution from another location (type G next to the instruction where you want to resume)

In fact, a special Breakpoint Primary Option menu displays when you press PF6 Menu at a breakpoint. This menu makes it easy to choose a breakpoint activity to accomplish your testing goal. When debugging your own programs, you typically perform one or more of these activities, which are detailed in the next article. Since PL1DEMO has no more errors, we will complete the demo session.

As you continue testing and debugging, you should clean up by removing breakpoints that are no longer needed so that when retesting the program it will not stop unnecessarily. Next, remove the unconditional breakpoint you just set so that PL1DEMO does not stop at this statement when it re-executes.

1. Overtyping U with X to remove the breakpoint.

```

CA InterTest for CICS - PROTSYM FILE UNCOND BEFORE BREAKPOINT
COMMAND ==>
Program= PL1DEMO  Option #      Stmt #                               Search=          Margin= 01
-----
----- TASKNUM                               | +00001.
-----+-----
-   639.1    1      IF TASKNUM > 2 THEN
-   640.1    1      GOTO END_SESSION;
X   ==>.1    1      IF TASKNUM = 1 THEN
-   642.1    1      GOTO SENDSCR3;
-   643.1    1      GOTO END_SESSION;
.
.
.
    
```

2. Press Enter.
CA InterTest for CICS removes the unconditional breakpoint.

Remove Breakpoints from the Status Display

You can also remove a breakpoint from the Status display:

1. Type **STATUS** on the command line.
2. Press Enter.
CA InterTest for CICS displays the Monitoring Status screen.

```
----- CA InterTest for CICS MONITORING STATUS -----
COMMAND ==>

Type + to expand or - collapse option levels displayed below,
or R to remove option(s).

  Option      Description              Attributes
  - PL1DEMO   Program monitor entry      PL/I
  |           Waiting at breakpoint      Task 00065, UBP since 12:10 p.m.
  - |-.ANY    User monitoring options     Active
  |           Symbolic listing file       PROTSYM
  r - |-.UBP   Unconditional breakpoint    #641
  |           Option ID              8F7A1400
  |           From, to terminals    U006, U006
  - |-.SLB    Source listing breakpoints  U006
  |           *** End of data ***

PF1 Help      2 Refresh    3 End        4 Return     5 Collapse   6 Expand
PF7 Backward  8 Forward    9            10           11           12
```

3. Type **R** to the left of the unconditional breakpoint (UBP) to remove it.
4. Press Enter.
An asterisk appears next to the breakpoint, indicating that a command has been processed for it.
5. Press PF2 to refresh the screen.
The unconditional breakpoint no longer displays.
6. Press PF3 to return to the Source Listing Breakpoint screen.

Resume Program Execution

1. Press PF5 to continue program execution.
CA InterTest for CICS resumes program execution. PL1DEMO displays a screen confirming that you successfully corrected the ASRA.
2. Press Enter.
PL1DEMO displays the following screen.

```
*****
****                                     ****
****                                CA InterTest Demo Session                ****
****                                     ****
*****
```

You have completed the sample CA InterTest test session. As part of this session, you:

- * displayed program source code and compiler output online
- * displayed and modified main storage
- * controlled program execution

This is just a fraction of CA InterTest's capabilities. You can also:

- * display or modify any CICS file, or DL/1, DB2, or SQL/DS database
- * set and remove many kinds of monitoring options

Press ENTER or CLEAR to complete the termination.

This screen reminds you that we have touched on just a few of CA InterTest for CICS's powerful, yet easy-to-use, testing and debugging facilities.
To complete this part of the sample test session, press Clear or Enter to clear your screen.

What You Learned

The demo session has taught you the basics of using CA InterTest for CICS to test a program.

You have learned how to perform the following tasks:

- **Select a source program for display**
Go to ITST Menu Option 1.1. Select a program from the list.
- **Set monitoring for a program**
Use the MONITOR command or PF5 from Source Listing.
- **View a Monitoring Status display**
Use the STATUS command or PF12 from Source Listing.
- **Interpret the information the application provides when it detects an error**
View the Automatic Breakpoint and press PF1 for help.
- **Examine the value of a data item in a Keep window**
View the data in the AutoKeep Display window
- **Modify main storage**
Overtyping the value in the Keep Window or type D to the left of a variable. Place the cursor under the variable and press Enter for a CORE Main Storage display. On the CORE screen, overtype the values and press Enter.
- **Set an unconditional "before" breakpoint**
Type U to the left of an instruction and press Enter.
- **Remove a breakpoint from the Monitoring Status display**
Use PF12 to view Monitoring Status. Type R next to a UBP and press Enter.
- **Remove a breakpoint from the Source Listing Display**
On Source Listing, overtype U with X and press Enter.

- **Remove a data item from a Keep window**
Type X to the left of a Keep window entry and press Enter.
- **Resume program execution**
Use the RESUME command or PF5.

Now read the next article to learn how to use these and other basic CA InterTest for CICS functions to test your own programs.

PL/I Debugging Programs

This article provides an overview of key CA InterTest for CICS functions. For complete information, see [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging) and the Help facility.

- [Basic Debugging Tasks Checklist \(see page 201\)](#)
- [Display Your Source Listing \(see page 202\)](#)
- [Start a Test Session \(see page 203\)](#)
- [Set and Remove Breakpoints \(see page 204\)](#)
- [Inspect and Modify Main Storage \(see page 208\)](#)
- [Keep Variables in the Keep Window \(see page 210\)](#)
- [Work with Arrays and Qualified Variables \(see page 212\)](#)
- [Inspect and Modify Auxiliary Storage \(see page 213\)](#)
- [Resume Program Execution \(see page 216\)](#)
- [Abend a Task \(see page 218\)](#)
- [Get Help \(see page 218\)](#)
- [End a Test Session \(see page 220\)](#)
- [Correct the Source Code \(see page 220\)](#)

Basic Debugging Tasks Checklist

The following list describes the debugging tasks:

- Display the source listing
- Set monitoring for the program
- Set breakpoints
- Set other monitoring options as needed
- Initiate the program
- End the test session

When the program is stopped at a breakpoint, you can do the following tasks:

- Inspect and modify main storage

- Inspect and modify auxiliary storage
- Set and remove breakpoints
- Keep variables in the Keep window to observe changes in their values
- Resume execution
- Abend the task

Display Your Source Listing

You can display your source listing online at any time from CICS through the ITST transaction or the LIST=*programe* transaction (where *program* is the program name). The Basic Demo explains how to display a source listing using the ITST transaction. For additional information on how to use LIST=, see [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging) .



Notes:

- The standard CA InterTest for CICS transaction IDs, such as LIST, ITST, CORE, and FILE, could have been changed at your site. If any of the transactions mentioned in this article do not function as described, contact the person who installed CA InterTest for CICS.
- Depending on your defaults, the options and PF keys available to you might not be displayed at the top of the Source Listing Display. If this information is not displayed, you can view it by pressing PF4 to access the Profile screen, and then setting Display window to T.

Display Sections of Your Program

The options (OPTS) available at the Source Listing screen make it easy to display any section of your PL/I program. However, all of the OPTS labels cannot be displayed at once. Tab to the More field and press Enter to view options not currently displayed.

The following table shows what program sections are displayed when you use the LOCATE command or option numbers:

Command	Option #	Section
L.DX	1	Data Name Cross Reference
L.AG	2	Aggregate Length Table
L.SR	3	Storage Registers
L.SS	4	Static Storage Map
L.VS	5	Variable Storage Map
L.OF	6	Offsets
L.GC	7	Generated Code (Assembler-like)

L.EM	8	Error Messages
L.PX	13	Procedure Cross-Reference
L.LX	14	Label Cross-Reference

Search for Strings and Labels

You can also search for and display a character string by typing a FIND string NEXT/PREV command on the command line or option 9 (Search forward) or 10 (Search backward) in the Option # field and typing the character string in the Search field. For example, the following screen shows how you would search backward for the string *custname*.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==> f custname prev
Program= PL1DEMO  Option #      Stmt #      Search=      Margin= 01
-----+-----
-                CONTINUE_TASK:
-                MAPNUM = '00';
-   95  1  0      TASKNUM = TASKNUM + 1;
-   96  1  0      IF TASKNUM > 2 THEN
-                   GOTO END_SESSION;
-   97  1  0      IF TASKNUM = 1 THEN
-                   GOTO SENDSCR3;
-   98  1  0      GOTO END_SESSION;

```

Start a Test Session

This section describes how to start a test session.

Turn on Monitoring

To test a program, you must instruct CA InterTest for CICS to monitor it. When CA InterTest for CICS monitors a program, it detects and prevents errors before they occur, including the following violations:

- All storage violations
- All CICS abends
- Any statement that would cause a program check or other abend
- All illegal or invalid instructions that would cause CICS to crash
- All wild branches
- All violations of CICS standards

The easiest way to turn on monitoring is from the program's source listing. Perform the following steps:

1. Display the source listing as described in [Display Your Source Listing \(see page 202\)](#). CA InterTest for CICS displays the Source Listing Display screen for the program.
2. Press PF5 to set monitoring, which remains in effect for the program until specifically removed.

Execute a Program

Once you have turned on monitoring for a program, you are ready to execute it. Perform the following procedure to execute the program:

1. Press PF3 to exit the Source Listing facility. If you entered Source Listing from an ITST menu, you are returned to that menu.
2. To exit the ITST menus, type =X in the top field and press Enter. The ITST transaction ends and you are returned to CICS.



Note: Before returning to CICS, you might want to set breakpoints directly on the source listing, as described in [Set and Remove Breakpoints \(see page 204\)](#).

3. On a clear CICS screen, type the transaction identifier of the program you are monitoring.

Once you type the program's transaction ID, one of the following occurs:

- Your program runs to completion. The results might not be correct.
- CA InterTest for CICS stops your program at a breakpoint -- either one set by you or one automatically triggered by CA InterTest for CICS because it detected an error.

Set and Remove Breakpoints

When you test a program, it is important to be able to halt program execution at specified locations. A halt in program execution is called a *breakpoint*.

What You Can Do at a Breakpoint

When your program is stopped at a breakpoint, you can do the following:

- Examine the source listing
- Inspect and modify main storage
- Inspect and modify auxiliary storage
- Set and remove breakpoints
- Examine the backtrace
- Keep variables in the Keep window to observe changes in their values
- Set and remove monitoring options
- Specify indirect commands
- Resume execution

- Abend the task

We have already discussed how you can examine your source listing. The other breakpoint activities are discussed in this and the next article.

Types of Breakpoints

There are five types of breakpoints for PL/I programs:

- **Automatic**
The program stops because CA InterTest for CICS detected and prevented an error.
- **Unconditional**
The program stops at the location you specify.
- **Conditional**
The program stops at the location you specify if a condition is met. Optionally, conditional breakpoints can be set to stop at any instruction if a condition is met.
- **Request**
The program stops at every CICS command or macro, or at certain CICS commands or macros, or at calls to DL/I, DB2, or software.
- **Single-step**
The program stops after executing one or more verbs.

An automatic breakpoint occurs when CA InterTest for CICS detects an error. When a program is stopped at an automatic breakpoint, you can either correct the error or go around it. You can press PF1 to find out what caused the error and how to use CA InterTest for CICS to fix it. You set all other breakpoints.

In this article we explain how to set and remove *unconditional* breakpoints, because you use these the most. For information on conditional and request breakpoints, see [PL/I Advanced Monitoring Features \(see page 220\)](#) . Single-stepping is discussed in [Resume Program Execution \(see page 216\)](#) .

The next sections explain how to do the following tasks:

- Set breakpoints at statements on the source listing
- Set breakpoints at all references to a data name
- Set breakpoints at all or selected paragraph names or labels

Set Unconditional Breakpoints at Statements on the Source Listing

To set an unconditional breakpoint on the program source listing, perform the following procedure:

1. Enter **U** or **)** in column 1 next to the statement where you want the breakpoint. The **U** sets a breakpoint that stops *before* the statement is executed. The **)** sets a breakpoint that stops *after* the statement is executed.
2. Press Enter.
The following screen shows how to set a "before" breakpoint.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>
Program= PL1DEMO Option # Stmt # Search= Margin= 01
OPTS 1 Data xref 2 Aggregate 3 Stor reqs 4 Stat stor 5 Var stor More: +
      6 Offsets 7 Gend code 8 Err msg 9 Srch fwd 10 Srch bwd
PFKS 1 Help 2 3 End 4 Profile 5 Monitor 6 Menu
      7 Backward 8 Forward 9 Next Wnd 10 11 12 Status
-----+-----
- 637.1 MAPNUM = '00';
- 638.1 1 TASKNUM = TASKNUM + 1;
- 639.1 1 IF TASKNUM > 2 THEN
- 640.1 1 GOTO END_SESSION;
u 641.1 1 IF TASKNUM = 1 THEN
- 642.1 1 GOTO SENDSCR3;
- 643.1 1 GOTO END_SESSION;
.
.
.

```

After you set a breakpoint, CA InterTest for CICS redisplay the screen with an uppercase U or a). This character remains until you remove the breakpoint. You can set breakpoints at any time; either before executing the program or when the program is stopped. Although where you decide to set breakpoints depends on the specifics of your program, you might want to set breakpoints:

- At the beginning of the program, so when the program executes you can set additional breakpoints
- At labels, so you can examine the contents of variables at the start of sections
- Before a CALL, so you can dynamically control the program path
- At each location named in an EXEC CICS HANDLE CONDITION, so you can verify error handling

To *remove* an unconditional breakpoint, overwrite the U with an X, or type an R next to the breakpoint's entry on the Monitoring Status display (PF12 from Source Listing). You want to remove breakpoints no longer needed so that when retesting the program, you are not unnecessarily stopped.

Set Unconditional Breakpoints from Cross-Reference Sections

The Cross Reference sections of the PL/I listing allow you to set many breakpoints at once. You can set unconditional breakpoints from the sections in the listing:

- Data Name Cross Reference Table (L .DX or Option 1)
- Procedures Cross-Reference (L .PX or Option 13)
- Labels Cross-Reference (L .LX or Option 14)

To display a section, type the command on the command line or the number in the Option # field on the Source Listing screen, as shown next, and press Enter.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==> l .dx
Program= PL1DEMO Option # Stmt # Search= Margin= 01
-----+-----

```

```

- 636.1          CONTINUE_TASK:
- 637.1          MAPNUM = '00';
- 638.1          1      TASKNUM = TASKNUM + 1;
- 639.1          1      IF TASKNUM > 2 THEN
- 640.1          1          GOTO END_SESSION;
U 641.1          1      IF TASKNUM = 1 THEN
- 642.1          1          GOTO SENDSCR3;
- 643.1          1      GOTO END_SESSION;
.
.
.

```

CA InterTest for CICS displays the Data Name Cross Reference Table section, as follows.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>
Program= PL1DEMO Option #      Stmt #      Search=      Margin= 01
-----+-----
u          ATTRIBUTE/XREF TABLE-
          LINE.FILE IDENTIFIER          ATTRIBUTES
          515.1   ADDR          BUILTIN
          Refs: 252.1 252.1 252.1 361+
                389.1 402.1 409.1 416+
                423.1 430.1 437.1 444+
                451.1 458.1 468.1 478+
                485.1 492.1 499.1 509+
                525.1 528.1 529.1 530+
                531.1 532.1 533.1 534+
                556.1 558.1 568.1 570+
                570.1 583.1 585.1 591+
                593.1 593.1 608.1 610+
                610.1 619.1 621.1 621+
                621.1 627.1 629.1 651+
                653.1 653.1 653.1 660+
                662.1 673.1 675.1 675+
                675.1 686.1 688.1 688+
                696.1 698.1 707.1 709+

```

Setting Unconditional Breakpoints at All Procedures and All Labels

To set unconditional breakpoints at all Procedure Names and All Labels, display the Cross Reference table using Option # 1 (or L .DX co mmand), type **U** or **)** next to the "ATTRIBUTE/XREF TABLE" heading line and press Enter. A message prompts you to confirm the request for setting many breakpoints by pressing PF3.

To set breakpoints at all Procedure names only, go to the Procedures Xref section using Option # 13 (or L .PX command), and then type **U** or **)** next to the "PROCEDURE ATTRIBUTE/XREF TABLE" header line and press Enter.

To set breakpoints at all Labels only, display the Labels Cross Reference section using Option # 14 (or L .LX command), and then type a **U** or **)** next to the "LABEL ATTRIBUTE/XREF TABLE" header line and press Enter.

You can remove all of these breakpoints at once by typing an X on the appropriate heading line and pressing Enter, or you can remove them individually from the Status d isplay (type R next to an entry) or one of the Cross Reference Sections of the Source Listing (type X).

Setting Unconditional Breakpoints at Data Items

From the Data Name Cross Reference Table section (Option 1) you can also set breakpoints next to individual data item s. Display the section, scroll forward using PF8, and type **U** or **)** next to the data item.

You can *remove* the breakpoints from either the Source Listing display (overtyping the U or) with X) or the Status display (typing R next to the breakpoint entry).

Inspect and Modify Main Storage

The ability to inspect main storage at a breakpoint means you can determine the values of variables without using a dump. It is easy to find logic errors because you can display the value of a variable where it is defined and at any point in the code where it is referenced. You can also dynamically change its value to correct errors or test other program paths.

Remember that changes to program storage are dynamic; that is, the change affects only the current test session. To correct program errors, you must change the source code and recompile the program.

There are many ways to inspect and modify main storage. We are going to discuss one of the easiest methods of viewing and changing program storage here. Keep Variables in the Keep Window explains another easy way to display and modify the values of variables.

Remember, you can inspect system storage at any time, independent of program monitoring and execution.

Display the Value of a Variable

When you are stopped at a breakpoint, you can request the display of the value of a variable directly from the source listing.

1. To display its value where it is *defined*, type D to the left of the statement defining it and press Enter, as shown in the following screen.

```
CA InterTest for CICS - PROTSYM FILE UNCOND BEFORE BREAKPOINT
COMMAND ==>
Program= PL1DEMO Option # Stmt # Search= Margin= 01
-----
----- DMAP000.DFHMS9 | .....
-----+-----
- 514.1 UNSPEC, SUBSTR,
- 515.1 LENGTH, ADDR) BUILTIN;
- 516.1 1 DCL 1 TS_AREA,
- 517.1 2 TS_SWITCH CHAR(1),
d 518.1 2 TASKNUM FIXED DEC(5),
- 519.1 2 TS_TEXT CHAR(32);
- 520.1 1 DCL (TS_ITEM,TS_LEN) FIXED BIN(15,0);
- 521.1 1 DCL REC_RBA CHAR(4) INIT(' ');
.
.
.
```

2. To display the value of a variable where it is *referenced*, type D to the left of the statement referencing it, place the cursor under any character in the variable, and press Enter. CA InterTest for CICS responds by displaying the contents of the variable in structured format, as shown next.

```
CA InterTest for CICS - MAIN STORAGE UTILITY - Termid = U057

Starting at Address =20304231 Structure Display Format
02 TASKNUM | ?00000. |
02 TS_TEXT | ..... |
           | ..... |
```

| |

```
-----
PF1 Help      2          3 End      4 Return    5          6 Dump
PF7 Backward  8 Forward   9 Caps Off 10          11 Redispl 12 Structure
CORE='TASKNUM'
CAIN0478 AUTOMATIC DECIMAL FIELD CONTAINS INVALID PACKED DATA
```



Note: CA InterTest for CICS displays more than just the contents of the specified variable (TASKNUM); it also displays all the items below it in the same PL/I structure.

3. Press Clear or PF3 to return to the source listing.

Modify the Value of a Variable

You can modify the value of a variable by over typing the bytes in the main storage display and pressing Enter. For example, you could change the contents of TASKNUM in the previous figure by overtyping the question mark (this indicates the field contains an invalid value for its type), with a 0 and hitting EOF (erase to end of field).

Modify the Value of a Variable with the MOVE Statement

Overtyping the main storage display is the easiest way to change the value of a variable. However, you will want to use the CA InterTest for CICS generated MOVE statement to modify a variable when you do not know the type of data (binary, packed, and so on) or its length. The MOVE statement takes care of all the details for you.

1. To modify a variable where it is defined, type M to the left of the statement defining it and press Enter. For example, to modify the value of TASKNUM, you would type M instead of D.
2. If you type M to the left of multiple lines, CA InterTest for CICS generates multiple MOVE statements.
3. To modify a variable where it is referenced, type M to the left of the statement referencing it, place the cursor under any character in the variable, and press Enter. CA InterTest for CICS generates and displays a fill-in-the-blanks MOVE statement, shown next.

```
CA InterTest for CICS - PROTSYM FILE UNCOND BEFORE BREAKPOINT
COMMAND ==>>
Program= PL1DEMO Option # Stmt # Search= Margin= 01
-----
_____ DMAP000.DFHMS9 | .....
-----+-----
MOVE _____
to
TASKNUM
```

Overtyping Underscores with a Data-Name, Figurative Constant, Alphanumeric Literal, or Numeric Literal

4. Type one of the following values in the MOVE field:
 - A variable name, such as NEXT_TASKNUM
 - One of the following keywords: ZEROS, SPACES, LOW-VALUES, HIGH-VALUES, QUOTES
 - An alphanumeric literal enclosed in single quotes, such as 'YES'
 - A numeric literal, with or without a leading plus, +, or minus sign, -, such as -8.
5. Press Enter to execute the MOVE statement.
CA InterTest for CICS displays a main storage display showing the new value of the variable. If you execute several MOVE statements, CA InterTest for CICS shows the main storage displays one-at-a-time. Press Clear to see the next one.
6. From the main storage display, press Clear or PF3 to return to the source listing.

Keep Variables in the Keep Window

The Keep window lets you display the values of an unlimited number of data items directly on the source listing. If more than six variables are selected for the Keep window, PF19 and PF20 are available to scroll forward and backward through the Keep window. Keeping data items in the window lets you observe changes in their values as the program executes. You also can change values by overtyping the displayed bytes.

The Keep window consists of two parts:

- *Dynamic* keep elements consist of the variables based on the current line of code. They are automatically displayed using the AutoKeep Display profile feature. They are shown in highlighted text.
- *Static* keep elements are those that remain in the Keep window for the duration of the execution of the program until you remove them. The *static* keep elements are displayed before the *dynamic* elements. Use scrolling to view all the elements.

You can change any *dynamic* element to a *static* element simply by following these rules:

When the Keep window is active, the options and PF key functions are not displayed. Press PF4 if you need to refer to them. When all the data items in the Keep window are removed, the options and PF key functions are redisplayed. A command line is available to help you perform CA InterTest for CICS functions when the Keep window is active.

Add a Variable to the Keep Window (static keep element)

You can add variables to the Keep window whenever your source listing is displayed. Identify the variables whose values you want to observe either before you execute the program, or when you are at a breakpoint.

Follow these rules to add a variable to the Keep window:

- To add a variable where it is *defined*, type K to the left of the statement defining it, and press Enter.
- To add a variable where it is *referenced*, type K to the left of the statement referencing it, place the cursor under any character in the item, and press Enter.

When you add a variable to the Keep window, CA InterTest for CICS responds by displaying the first 12 bytes of the variable in hexadecimal and character format, as shown next.

```
CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= PL1DEMO  Option #      Stmt #                      Margin= 01
                        Search=
-----
----- EIBTRNID                |  DEMP
----- EIBTRMID                |  U057
----- TASKNUM                 |  ?00000.
-----+-----
- 636.1          CONTINUE_TASK:
- 637.1          MAPNUM = '00';
A ==>.1         1      TASKNUM = TASKNUM + 1;
- 639.1         1      IF TASKNUM > 2 THEN
- 640.1         1          GOTO END_SESSION;
- 641.1         1      IF TASKNUM = 1 THEN
- 642.1         1          GOTO SENDSCR3;
- 643.1         1      GOTO END_SESSION;
.
.
.
```

Display Variable Structure

To display the entire PL/I structure of a variable, type D to the left of the variable in the window and press Enter.

```
CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= PL1DEMO  Option #      Stmt #                      Margin= 01
                        Search=
-----
----- EIBTRNID                |  DEMP
----- EIBTRMID                |  U057
d----- TASKNUM                 |  ?00000.
-----+-----
- 636.1          CONTINUE_TASK:
- 637.1          MAPNUM = '00';
A ==>.1         1      TASKNUM = TASKNUM + 1;
- 639.1         1      IF TASKNUM > 2 THEN
- 640.1         1          GOTO END_SESSION;
- 641.1         1      IF TASKNUM = 1 THEN
- 642.1         1          GOTO SENDSCR3;
- 643.1         1      GOTO END_SESSION;
.
.
.
```

CA InterTest for CICS responds by displaying the variable and all the items below it in the same structure.

Modify the Value of a Variable

Modify the value of a variable in the Keep window by performing the following procedure:

1. Overtyping the displayed bytes, as you would overtype the bytes in a main storage display
2. Type M to the left of the variable to use the CA InterTest for CICS MOVE statement. Both methods of modifying a variable are discussed in [Inspect and Modify Main Storage \(see page \)](#).



Note: If you try to change password protected storage areas not owned by your program, CA InterTest for CICS prompts you to enter the password.

Remove a Variable from the Keep Window

To remove a variable from the Keep window, type X to the left of the variable. In this example, two variables (EIBTRNID and EIBTRMID) are removed from the Keep window.

```
CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= PL1DEMO  Option #      Stmt #                      Margin= 01
                    Search=
-----
x____ EIBTRNID                |  DEMP
x____ EIBTRMID                |  U057
____  TASKNUM                 |  ?00000.
-----
-   636.1                    CONTINUE_TASK;
-   637.1                    MAPNUM = '00';
A  ==>.1      1              TASKNUM = TASKNUM + 1;
-   639.1                    IF TASKNUM > 2 THEN
-   640.1                    GOTO END_SESSION;
-   641.1                    IF TASKNUM = 1 THEN
-   642.1                    GOTO SENDSCR3;
-   643.1                    GOTO END_SESSION;
```

When all variables are removed, the Keep window no longer displays. If your Profile had Display window set to T, the Option and PF key title lines are redisplayed.

Work with Arrays and Qualified Variables

This release provides the ability to view or modify the following:

- Array elements, both structured and non-structured
- Qualified variables within a procedure, block or program with up to 15 levels of qualification

Array Elements

You have the ability to view or change elements of arrays that are up to 15 dimensions deep.



Note: Arrays with unknown boundaries at compile time are not supported.

When you use the D or K line commands from Source Listing, place the cursor under the element and CA InterTest for CICS automatically picks up the dimensions.

When you are entering an array element for a formatted move command, or CNTL or CORE command, use the syntax:

```
FIELD(A,B,C)
```

To modify the storage of an array element using a CORE command, use the syntax:

```
CORE=MOVE 1 TO 'FIELD(A,B,C)'
```

Qualified Variables

You can view or change elementary variables even if they are defined as the same name in different procedures or blocks using qualified variable syntax in a CNTL or CORE command or Source Listing Search criteria.

```
CORE=MOVE 1 to 'procname:Q1.Q2.Q3.Q4.Q5.Q6.Q7'
```

If you are using the M line command from the Source Listing facility and select a non-unique variable, the generated MOVE command qualifies the selected variable with the Procedure name.

Inspect and Modify Auxiliary Storage

You can use CA InterTest for CICS to inspect and modify the following storage:

- VSAM and BDAM files
- DL/I, DB2, and SQL/DS databases
- Temporary storage
- Transient data

You can inspect and modify auxiliary storage at any time, regardless of whether a program is executing. This capability lets you maintain files and databases *without* writing one-time programs. You can perform any VSAM, BDAM, or DL/I function and most SQL functions. The ability to inspect and modify auxiliary storage when a program stops at a breakpoint lets you change test data in the middle of a test session. Any changes you make are permanent; that is, changes to the file remain after the test session ends.

Inspect Auxiliary Storage

1. To access the auxiliary storage facility, select 4 Auxiliary storage from the CA InterTest for CICS Primary Option Menu, and then specify the file you want to examine on the Auxiliary Storage Menu.

```
----- CA InterTest for CICS AUXILIARY STORAGE MENU -----
-----
OPTION ===> 1

Select an auxiliary storage type, specifying optional criteria below.

1 Files - Display/select files for access
2 DB2 database - Invoke DB2 SQL interface facility
3 DL/I database - Access DL/I database
4 TD queues - Display/select transient data queues for access
5 TS queues - Display/select temporary storage queues for access

Type specific or generic file/queue name(s):
```

CA InterTest™ and CA SymDump® - 11.0

(Valid mask characters are * and/or +)

```
file_____
.
.
.
```

- Alternatively, if you are at a clear CICS screen, you can invoke the FILE transaction. CA InterTest for CICS displays the following screen:

```
DATATYPE= FC FILEID=          MODE=      LOG=ON  TODEST=      PASSWORD=
FUNC=     SUBFUNC=          RETMETH=   ARGTYP=   SRCHTYP=
MESSAGE=
  RETNRCID=                  CHGLEN=
  RCID=
  DATA=                      SIZE= 0000
FORMAT= D 00112233 44556677 8899AABB CCDDEEFF *0123456789ABCDEF*
LOC 0000 .....
```

CA InterTest for CICS V10.0

Copyright ©
2016 CA. All rights reserved.

```
-----
1 Help      2 Format C  3 End      4 BEGB     5          6 DataType DL
7 Page bwd  8 Page fwd  9 Caps Off 10 Top     11 Bottom  12
```



Note: If the FILE transaction does not display the previous screen, it is probably because the person who installed CA InterTest for CICS changed the name of the FILE transaction. Contact that person to find out the correct name.

After you enter the file or database and, optionally, the record or segment, CA InterTest for CICS displays the data. A sample VSAM record in dump format follows.

```
DATATYPE= FC FILEID= INVNTRY  MODE=      LOG=OFF  TODEST=      PASSWORD=
FUNC=     SUBFUNC=          RETMETH=   ARGTYP=   SRCHTYP=
MESSAGE= RECORD OBTAINED FOR VIEWING

RETNRCID=F0F0F0F0F0F0F0F5          CHGLEN=
  RCID=C`000000005`
  DATA=                              SIZE= 0050
FORMAT= D 00112233 44556677 8899AABB CCDDEEFF *0123456789ABCDEF*

LOC 0000 .....
  0000 F0F0F0F0 F0F0F0F0 F0F54040 0087338F 00000000005 DSORG=VSKS
      0010 F0F3F9F9 0407835C 40404040 40404040 0399... RECFM=VB
      0020 40404040 40404040 C1C2C340 C3D6D9D7 ABC CORP LRECL=0050
      0030 F0F9F1F2 F8F74040 0000598C 40404040 091287 BLKSIZE=0000
      0040 D1D6C3D5 404040E2 D4C9E3C8 40404040 JOHN SMITH KEYPOS=0000
      0050 KEYLEN=0A
      0060 STRNO=01
      0070
```



```

        RCID= C'1'
        DATA=                                     SIZE= 02D8
        FORMAT= S USE= PRODUCT.PRODREC
        LOC 0050          Name                    Hexadecimal          Character
-----
01 PRODREC
02 PRODUCT-TYPE          D3
02 PRODUCT-CODE         C2D4D5C8 40404040
02 PRODUCT-DESCRIPTION  E6D9C5D5 C3C8
02 PRODUCT-SITE         D1
-----

1 Help      2 Format D   3 End      4 BEGB     5
7 Page bwd  8 Page fwd  9 Caps Off 10 Top     11 Bottom  12
6 DataType DL

```

4. Structured format lists the contents of each data name in both hexadecimal and character format. To modify the record or segment, overwrite either the hexadecimal or character data and press Enter. To rewrite the modified record, type PUT in the FUNC= field and press Enter.

5. When a program is stopped at a breakpoint, you can substitute an asterisk (*) for the symbolic-name. For example, if program PAYROLL is stopped at a breakpoint and you type:
 USE=*.payrec

CA InterTest for CICS uses the program named PAYROLL and the PAYREC structure.

6. To begin the structure with a specific data name, specify the following command in the DATA field:

```
FIND=data-name
```

Where *data-name* is the data name to be displayed.

Resume Program Execution

When you are finished performing breakpoint activities such as displaying and modifying main or auxiliary storage, you can resume program execution. Perform the following procedures:

1. Resume execution from a specific location
2. Single-step execution one verb at a time

See the End a Test Session for information on the status of your program after it completes execution.

Resume Execution from a Specific Location

To resume program execution from the breakpoint location where your program stops, the Source Listing screen must display.

1. Press PF3 from any CA InterTest for CICS screen until the breakpoint screen appears.

- Press PF5 to continue from the current breakpoint. You can also type RESUME on the Command line and press Enter.
Execution continues until the program halts at another breakpoint or the task completes.
If you are stopped at an *automatic* breakpoint, you must first correct the error that triggered the breakpoint before continuing execution from that point, or you can go around the error and continue execution from another location.

To resume program execution from another location, perform the following steps:

- Display the statement where you want to resume execution. For details, see [Display Your Source Listing \(see page 202\)](#) .
- Type G to the left of that statement.
- Press Enter. Program execution resumes from that statement.
The following example shows how to resume execution from a location prior to an automatic breakpoint. This is useful if you corrected a record or dynamically modified an incorrect file name and now want the program to re-read the record or file.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= PL1DEMO  Option # 9      Stmt #                               Margin= 01
                               Search= READATA
-----
-----+-----
TASKNUM                        | ?00000.
-----+-----
- 1308.1      1                GOTO SENDMP00;
- 1309.1
g 1310.1      1      READATA:
- 1311.1                /* EXEC CICS HANDLE CONDITION  DSIDERR
- 1312.1                ERROR      (GENERR)
- 1313.1                NOTOPEN  (NOOPEN) *
- 1314.1                DO;
.
.
.

```

- When you press Enter, execution resumes from READ-DATASET, the location you identified with G.

Single-Stepping

Single-stepping lets your program execute one COBOL verb and then stop again. You can adjust the step amount so that the program executes a specified number of verbs between stops.

To single-step, type NEXT in the command line and press Enter or press PF10 from the source listing.

Your program executes the next verb and then stops.

To single-step from a different program location perform the following steps:

- Display the statement where you want to resume execution. For details, see [\(see page 202\) Display Your Source Listing \(see page 202\)](#) .
- Type G to the left of that statement.
- Type NEXT in the command line and press Enter or press PF10.

To adjust the step amount, perform the following steps:

1. Press PF4 to access the Source Listing Profile screen.
2. Change the number in the Stepping amount field.
3. Press Enter.
CA InterTest for CICS changes the step amount and redisplay the source listing. If Titles are set to on (an option on the Source Listing Profile screen), the new step amount appears next to PF10 at the top of the Source Listing Breakpoint screen.

Abend a Task

When your program is stopped at a breakpoint, you can abend your task rather than resume execution.

1. Type ABEND (or, =3) in the Command line of any Breakpoint display.
CA InterTest for CICS displays the following screen.

```
----- CA InterTest for CICS ABEND BREAKPOINTED TASK -----
-----
COMMAND ===>
```

Type an abend code, then press ENTER.

```
Abend Code   ____   Abend code options are:
                blanks       Normal abend, no dump
                XXXX         Abend exits cancelled, no dump
                your code    Your abend code, dump taken
                .
                .
                .
```

2. To cancel the abend, press PF3.
CA InterTest for CICS redisplay the Primary Option Menu.
3. To abend *without* a dump:
 - a. For a normal abend, press Enter.
 - b. To cancel all program exits, specify code XXXX and press Enter.
4. To abend *with* a dump, specify a four-character dump code and press Enter. The dump code you specify cannot be XXXX and cannot begin with the letter A.
For information on the status of your program after abending, see [End a Test Session \(see page 220\)](#).

Get Help

Help is always available when you are using CA InterTest for CICS. You can also use Help to initiate CA InterTest for CICS. Type Help on a clear screen under CICS.

Get Help in Using CA InterTest for CICS

When you are using CA InterTest for CICS, you can press PF1 at any time for information about CA InterTest for CICS functions. For example, the following screen lists the Help topics for the Source Listing facility.

```

CA INTERTEST - INTERACTIVE HELP FACILITY -
TUTORIAL: SOURCE LISTING FACILITY - FOR PL/I PROGRAMS

05 Set Monitoring
10 Locating areas in the listing
20 Search for a Character string
25 Setting Indirect Commands
30 PF key definitions
32 Primary/Line Commands
35 Set the Display Margins
40 Set/Remove Breakpoints
45 Set/Remove other monitoring Options
50 Displaying data items
55 Modify the value of data items
60 Resume TASK execution
65 Abending your TASK
70 PROFILE, AUTOSTEP, AUTOKEEP Options
75 Code Coverage Option
80 View Program Backtrace

-----
                                         (end)
-----
ENTER A SELECTION CODE FROM THE MENU, OR N FOR NEXT PAGE, P FOR PRECEDING PAGE,
M FOR RETURN TO PREVIOUS MENU, CLEAR TO EXIT, OR MESSAGE NUMBER. ==> 05

```

Help for Line Commands

Notice that selection code 32 lists Line Commands. These are the simple commands that you can enter on the Command line of a Source Listing Display, Breakpoint Display, or CA SymDump Breakpoint display, and include the commands abend, monitor, and status.

Get Help to Correct an Error

Whenever your program stops at an automatic breakpoint, CA InterTest for CICS displays a short message identifying the reason for the breakpoint. You can request help by pressing PF1 for information on what caused the error that triggered the breakpoint and how to fix it. The following screen explains how to correct an AEIL abend.

```

CA INTERTEST - INTERACTIVE HELP FACILITY -
TUTORIAL: ERROR MESSAGE AEIL

The dataset name referred to in the DATASET option cannot be found in the FCT.
Your program did not have a Handle Condition for this error. Or a second Handle
Condition without a routine for this error superseded the one that had a
routine for this error.

WHAT YOU CAN DO: If the named file was entered incorrectly and the one you
wanted exists, you may use the Replace File Option to dynamically replace the
file name and then use the resume task facilities to execute the CICS request
again. To perform the above functions from the Source Listing Breakpoint screen
you would:

1. Key =20s in the Option # field and press ENTER.
2. Tab to Replace file name: Key in the incorrect file name.
3. In the next field, key in the correct file name and press ENTER.
4. Press CLEAR to return to your Source Listing Breakpoint screen.
5. Key in G to resume execution at the beginning of the EXEC CICS command
and press ENTER.

-----
                                         (continued)
-----
ENTER N FOR NEXT PAGE, P FOR PRECEDING PAGE, F FOR FIRST PAGE, OR -
M FOR RETURN TO PREVIOUS MENU, CLEAR TO EXIT, OR MESSAGE NUMBER. ==> N

```



Note: When you leave and then return to an automatic breakpoint screen, the error message may no longer appear. For help on the cause of the error, press Clear to redisplay the error message before pressing PF1.

End a Test Session

After your program completes execution or you abend your task, you are returned to CICS.

The following table shows the status of your program after testing:

Program Area	Status after Testing
program source code	not modified
file/database updates by the program	permanent
file/database updates you made	permanent

CA InterTest for CICS monitoring, breakpoints, and other options set for the program remain in effect until they are specifically removed.

Correct the Source Code

At some point you might want to correct the source code and continue testing from a clean version of the program.

After you modified the source code and recompiled the program, you can load the new copy of the program by using the New Program Copy option on the Program Monitoring menu (ITST 2.1). The New Program Copy option resets symbolic breakpoints and other monitoring options for the recompiled program. It also resets the PPT entry to the program's new library address.

PL/I Advanced Monitoring Features

CA InterTest for CICS has many advanced features that help you detect and correct errors. In fact, whatever your specific testing needs, CA InterTest for CICS has the necessary tools.

This article describes just a few of the more commonly used features. For information about all CA InterTest features, see [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging) and the Help facility.

- [Set Options from the Monitoring Menus \(see page 221\)](#)
- [Set and Remove Conditional Breakpoints \(see page 222\)](#)
- [Request Breakpoints \(see page 224\)](#)
- [The Backtrace Facility \(see page 226\)](#)
- [Set the Code Counting Coverage Option \(see page 229\)](#)
- [Replacement, Protection, and Special Options \(see page 235\)](#)
- [Composite Module Testing \(see page 236\)](#)

- [Dump Analysis with CA SymDump \(see page 237\)](#)

Set Options from the Monitoring Menus

The monitoring menus of CA InterTest are the easiest way to set many monitoring options.

1. Access these menus from the Primary Option Menu by selecting 2 Monitoring:

```

----- CA InterTest for CICS PRIMARY OPTION MENU -----
OPTION ==> 2

    1 Source           - Display/select program source files/listings
    2 Monitoring       - Display/modify CA InterTest monitoring/activity
    3 Main storage     - Display/modify CICS storage areas
    4 Auxiliary storage - Display/access databases/files/queues
    5 Dump analysis    - Invoke CA SymDump CICS dump/trace capture facility
    6 Product help     - Invoke CA InterTest product help facility
    7 Status/Maintenance - Product status and maintenance functions
    8 What's new?      - Display information about CA InterTest for CICS
    X Exit             - Terminate menu processing
.
.
.
    
```

The Monitoring Menu appears, from which you can select a wide variety of different monitoring options. For a detailed explanation of each of these options, use the online help or [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging).

```

----- CA InterTest for CICS MONITORING MENU -----
OPTION ==>

    1 Programs         - Display/modify program monitoring options
    2 Transactions     - Display/modify transaction monitoring options
    3 Terminals        - Display/modify terminal monitoring options
    4 Status           - Display/remove current monitoring options
    5 Active tasks     - Display/purge active monitored tasks
    6 System-wide      - Display/modify global system-wide options
.
.
.
    
```

Option 1 Programs accesses the Program Monitoring Menu, where you can set and remove any option for a program, and request a status display.

The Program Monitoring menu is probably the most frequently used menu during a testing session.

```

----- CA InterTest for CICS PROGRAM MONITORING -----
-----
COMMAND ==>

    Type information and S to set or R to remove option(s) below.

    Program . . PL1DEMO_      Program name (or .ALL, .OPTIONS or generic)
    User ID  . . _____   User (or .ANY) for whom the program is monitored

    Option  Description                                                    More: +
    - Status      Display and/or remove monitoring options (S only)
    - Monitor     Monitoring (R removes monitoring and all options previously set)
    - UBP         Unconditional breakpoints (specific program only)
    - CBP         Conditional breakpoints (specific program only)
    - RBP         Breakpoints for CICS, DB2, DL/I or external CALL requests
    - Stmt Trace  Statement tracing and data monitoring (COBOL only)
    - New copy    Fetch new copy of program and reset monitoring options (S only)
    - Commands    Indirect commands defined for a specific COBOL or PL/I program
    - Replace     CICS resource name replacement options
    - Protect     Storage protection monitoring options
    
```

_ Special	Other options (storage allocation, file updating, etc.)				
_ Composite	Monitor multi-CSECT program's separately compiled components				
PF1 Help	2	3 End	4 Return	5	6
PF7 Backward	8 Forward	9	10	11	12

Quick access to this menu is available before or during a session, as described in the following table.

2. Type the fastpath entry in the indicated field and press Enter. The following list contains the fastpath entries that you can use to access the Program Monitoring menu:
 - CICS, clear screen: ITST 2.1
 - Primary Option Menu, Option field: 2.1
 - Source Listing display, Command field: =2.1
 - Breakpoint display, Command field: =1.2.1
3. After processing any program options from this menu, press PF3 to exit back to your test session. If you entered directly from CICS, you will exit back to the ITST Primary menu. From a Source Listing or breakpoint, PF3 takes you back to that display.

Set and Remove Conditional Breakpoints

Conditional breakpoints are breakpoints that take effect when a specific condition is met, such as when a counter exceeds a value. These breakpoints can help you isolate complex problems.

To set a conditional breakpoint, type C to the left of the statement where you want to set the breakpoint, as shown in the following screen, and press Enter.

```
CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ===>
Program= PL1DEMO Option # Stmt # Search= Margin= 01
-----+-----
- 599.1 1 GOTO SENDSCR1;
- 600.1
- c 601.1 1 WRITETSQ:
- 602.1 TS_SWITCH = ' ';
- 603.1 /* EXEC CICS WRITEQ TS QUEUE (TSQ_NAME)
- 604.1 LENGTH (TS_LEN)
- 605.1 FROM (TS_AREA)
- 606.1 MAIN */
```

CA InterTest for CICS then displays the menu on which you define the condition. In the following example, the breakpoint halts program execution only if TS_SWITCH is equal to 3.

```
CA-InterTest MONITORING COMMAND BUILDER - CONDITIONAL BREAKPOINT

Enter LEFT SIDE of condition (select one):
Data Name ts-switch -----
Special keywords: ----- Register # __ COBOL BLL # __

Enter OPERATOR (EQ, NE, GT, LT, GE, LE): eq Length: Left ___ Right ___

Enter RIGHT SIDE of condition (select one):
Data Name -----
Special keywords: ----- Register # __ COBOL BLL # __
```

```

Literal 3 _____
Optional offset: Enter + - @ OR % followed by a value.
Left side _____ Right side _____

_ ENTER S to Drop monitoring on a true condition

For location:
      601.1      1      WRITETSQ:

Use Help or documentation for use of special keywords
PF1 Help      2      3 End      4 Return      5      6
PF7           8      9      10      11      12
    
```

Conditional breakpoints are indicated by a C next to the line of code in your Source Listing.

You can easily remove a conditional breakpoint from where it displays in your Source Listing or from the Monitoring Status report.

Remove Conditional Breakpoints from the Source Listing

To remove a conditional breakpoint from your Source Listing, overtype the C with an X and press Enter, as shown in the following screen:

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>
Program= PL1DEMO Option # 9 Stmt # Search= WRITETSQ Margin= 01
-----+-----
_ 599.1 1 GOTO SENDSCR1;
_ 600.1
x 601.1 1 WRITETSQ:
_ 602.1 TS_SWITCH = ' ';
_ 603.1 /* EXEC CICS WRITEQ TS QUEUE (TSQ_NAME)
_ 604.1 LENGTH (TS_LEN)
_ 605.1 FROM (TS_AREA)
_ 606.1 MAIN */
.
.
.
    
```

Remove Conditional Breakpoints from the Status Report

Alternatively, you can always remove any breakpoint from the Monitoring Status report. To remove this conditional breakpoint from your program, perform the following procedure:

1. Type STATUS on the Command line to display the Monitoring Status report for the current program.
2. Tab to the entry for the Conditional Breakpoint (CBP option) you want to remove, type R to the left of its entry, and press Enter.

```

----- CA InterTest for CICS MONITORING STATUS -----
COMMAND ==>

Type + to expand or - collapse option levels displayed below,
or R to remove option(s).

Option      Description      Attributes
- - PL1DEMO Program monitor entry PL/I
- - |-.ANY User monitoring options Active
r - | Symbolic listing file PROTSYM
| -CBP Conditional breakpoint #601
| Option ID 0EC11E65
| Condition CORE=IF='TS_SWITCH'.EQ.C'3'
    
```

```

      |          Compare type          CORE command comparison
      |          From, to terminals    U057, U057
-    | -SLB    Source listing breakpoints U057
      |          *** End of data ***
  
```

```

PF1 Help      2 Refresh      3 End          4 Return      5 Collapse    6 Expand
PF7 Backward  8 Forward      9             10           11           12
  
```

CA InterTest for CICS places an asterisk to the left of the breakpoint to indicate that the removal was processed.

3. Press PF3 to return to your Source.

Request Breakpoints

Request breakpoints are breakpoints that take effect prior to CICS commands and macros and other program calls, such as calls to DL/I or DB2. With one specification, you can set breakpoints at all CICS commands or at specific commands, such as all REWRITE commands.

Use request breakpoints to stop a program before the following commands:

- Every CICS command or macro
- Specific types of CICS commands, such as all File Control or Program Control commands
- Specific commands, such as all READ or WRITE commands

Set Request Breakpoints

To set request breakpoints, perform the following steps:

1. Select option 2.1 Program Monitoring on the Primary Option Menu.

```

----- CA InterTest for CICS  PROGRAM MONITORING -----
-----
COMMAND ==>>

      Type information and S to set or R to remove option(s) below.

      Program . . PL1DEMO_      Program name (or .ALL, .OPTIONS or generic)
      User ID . . _____    User (or .ANY) for whom the program is monitored

      Option      Description                                          More:
+
- Status        Display and/or remove monitoring options (S only)
- Monitor       Monitoring (R removes monitoring and all options previously set)
- UBP           Unconditional breakpoints (specific program only)
- CBP           Conditional breakpoints (specific program only)
- *RBP          Breakpoints for CICS, DB2, DL/I or external CALL requests
- Stmt Trace    Statement tracing and data monitoring (COBOL only)
- New copy      Fetch new copy of program and reset monitoring options (S only)
- Commands      Indirect commands defined for a specific COBOL or PL/I program
- Replace       CICS resource name replacement options
- Protect       Storage protection monitoring options
- Special       Other options (storage allocation, file updating, etc.)
- Composite     Monitor multi-CSECT program's separately compiled components

PF1 Help      2             3 End          4 Return      5             6
  
```

PF7 Backward 8 Forward 9 10 11 12

2. In the Program field, type the name of the program for which you want to set request breakpoints.
3. Type S to the left of the RBP option (request breakpoints). Press Enter. The Request Breakpoint Selection menu appears. Using this menu specify the type of request breakpoint, such as all CICS commands, File Control and Task Control.

CA InterTest MONITORING COMMAND BUILDER - REQUEST BREAKPOINT SELECTION 13

```

Set one or more types of Request Breakpoints in:
PROG=PL1DEMO

_ ALL commands          _ DL/I   _ DB2   _ CALLS
- Address, Assign,     - Storage Control  - BMS
  Handles, Push, Pop  - Program Control - Trace Control
- Terminal Control    - Interval Control - Dump Control
- File Control        - Task Control     - Batch Data Interchange
- TD Control          - Journal Control  - Built-In Functions
- TS Control          - Syncpoints       - Sys Prog Functions
- Web access          - Business Trans   - 3270 Bridge

Enter 'n' to stop only every n'th time:
Term ID (or .ANY or .NO) where breakpoints will take effect:  ----
Term ID (or .ANY) that will receive the breakpoints:         ----
Statement no. of indirect command(s) to be executed:         ----
User ID (or .ANY) who will execute the program:              -----

PF1 Help      2          3 End      4 Return   5          6
PF7           8          9         10         11         12
    
```

Subsequent menus prompt you for the specific command, such as all File Control commands, READ, WRITE and REWRITE.

4. Type X in front of each type of command for which you want to set request breakpoints. After setting your request breakpoints, PF4 returns you to the Primary Option Menu. If you came from a Source Listing, PF3 returns you to your Source.

Remove Request Breakpoints

To remove request breakpoints, perform the following steps:

1. From the Source Listing display or breakpoint display, type STATUS on the Command line and press Enter. CA InterTest for CICS displays the Monitoring Status display showing all options for the current program.
2. Type R to the left of the request breakpoint (RBP) you want to remove and press Enter. CA InterTest for CICS places an asterisk to the left of the breakpoint to indicate that the command has been processed.
3. Press PF3 to return to the Source Listing or Breakpoint display.

The Backtrace Facility

When your program stops at a breakpoint, you might find it useful to examine the Backtrace Summary and the Source Listing Backtrace. The Backtrace facility shows the logic flow of the program and explains how processing reached this point.

The Backtrace facility enables you to do the following tasks:

- View the Backtrace Summary screen for a (high-level) summary of the program's execution
- Use special PF keys to step through the program's execution path
- Reposition the source listing to view the program's execution path from a different backtrace position

In addition to the Backtrace PF keys, all other Source Listing options, such as the Keep window, setting and removing breakpoints, and the ability to scroll backward and forward from the current source listing position are available from the Source Listing Backtrace screen.

Access the Backtrace Summary

To view the Backtrace Summary, press PF11 from the Source Listing Breakpoint screen, or, type BTRACE in the Command line and press Enter.

CA InterTest for CICS positions the Backtrace Summary Screen at the last executed statement, as follows.

CA InterTest for CICS - BACKTRACE SUMMARY

Program= **PL1DEMO** From 0002 To **0014** Of **0014**

Specify **S** then **ENTER** to display Source Listing BACKTRACE

```
-----
PFKS 1 Help      2 Backtrace 3 End      4          5 1st Stmt  6 Last Stmt
      7 Backward 8 Forward  9 Next Wnd 10         11 Prev Bloc 12 Next Bloc
-----
S Bkmk      Stmt Block | Source Listing
-----
  #19 ... #19 |          PL1DEMO: PROC(DFHEIPTR,COMMAREA_PTR) OP
  #19 ... #19 |          PL1DEMO: PROC(DFHEIPTR,COMMAREA_PTR) OP
  #19 ... #19 |          PL1DEMO: PROC(DFHEIPTR,COMMAREA_PTR) OP
  #525 ... #541 |          1  START_OF_PGM:
  #544 ... #544 |          1  IF EIBAID = DFHPF2 |          /**
  #548 ... #548 |          1  IF TS_SWITCH = DFHPF5 THEN
  #555 ... #555 |          1  DO;
  #561 ... #561 |          1 1  END;
  #573 ... #573 |          1 1  END;
  #578 ... #580 |          1  IF EIBCALEN > 0 |
  #588 ... #588 |          1 2  END;
  #595 ... #595 |          1 2  END;
  * #635 ... BKPT |          1  NOPROB:
CAIN2987 Last Backtrace Stmt Block
```

Read the Backtrace Summary

The Backtrace Summary screen summarizes the program's execution. This screen displays the statement number and source of the first statement in each contiguous group of executed program statements.

- The order of the statements reflects the program's execution path.

- The top statement is the first statement executed; the bottom statement is the most recently executed statement.

From the Backtrace Summary screen, you can do the following tasks:

- Reposition the display to view statement-by-statement details of the program's execution path by:
 - Pressing PF2
 - Marking a statement block with **s**, then pressing Enter
- Assign a *bookmark*, consisting of one to four characters, to one or more backtrace positions for faster and easier navigation through the Backtrace Summary
- Trace program execution using:
 - **PF5 1st Stmt**
Repositions the Backtrace Summary screen to the first or oldest entry in the Backtrace.
 - **PF6 Last Stmt**
Repositions the Backtrace Summary screen to the last or newest entry in the Backtrace.
 - **PF7 Backward**
Scrolls the Backtrace Summary screen backward (up) a full screen.
 - **PF8 Forward**
Scrolls the Backtrace Summary screen forward (down) a full screen.
 - **PF11 Prev Bloc**
Displays the previous statement block.
 - **PF12 Next Bloc**
Displays the next statement block.

Access the Source Listing Backtrace

To display the Source Listing Backtrace, press PF2 or type S next to a statement block in the Select column of the Backtrace Summary screen as shown in the previous screen.

The next screen shows the Source Listing Backtrace screen that displays.

```

CA InterTest for CICS  - PROTSYM FILE  ABEND DETECTED BACKTRACE
COMMAND ==>>
Program= PL1DEMO  Option #          Stmt #          Search=          Margin= 01
-----
----- TASKNUM          | ?00000.
===== Backtrace at #000635 (000635 -> 000638 executed 1 times) <=====
 633.1      1 1      END;
- 634.1
- ==>>.1      1      NOPROB:
- 636.1      CONTINUE_TASK:
- 637.1      MAPNUM = '00';
A 638.1      1      TASKNUM = TASKNUM + 1;
- 639.1      1      IF TASKNUM > 2 THEN
- 640.1      1      GOTO END_SESSION;
- 641.1      1      IF TASKNUM = 1 THEN
- 642.1      1      GOTO SENDSCR3;
- 643.1      1      GOTO END_SESSION;
- 644.1

```

```

- 645.1      1      SENDSCR3:                               /** ASRA CONGRATULAT
- 646.1                               /* EXEC CICS SEND MAP( 'DASRA' )
- 647.1                               MAPSET( 'IN25PMP' )
- 648.1                               WAIT
- 649.1                               ERASE */

```

Read the Source Listing Backtrace

To clearly detail the program's execution path, statement by statement, source statements that were executed from the current backtrace position are highlighted. When you specify a non-backtrace Source Listing option or use PF7 or PF8 to reposition the Source Listing, the highlighting of executed statements temporarily suspends until a backtrace-related PF key (next block, previous block, and so on) is entered.

When in Source Listing Backtrace mode, the current backtrace position always displays on the line just above the first source statement. This line displays as follows:

```

=====>BACKTRACE AT #nnnnn (nnnnn -> nnnnn executed nnnnn times)<=====

```

The information in this line indicates the following information:

- **#nnnnn**
Specifies where the program statement or offset the backtrace is currently positioned.
- **nnnnn -> nnnnn**
Specifies the first and last number of the statement block.
- **nnnnn times**
Specifies the number of times a statement block was executed.

On the Source Listing Backtrace, each relevant backtrace position is indicated in the listing by an arrow or a line:

- **====>**
Indicates the first statement in a statement block
- **<====**
Indicates the last statement in a statement block
- **====**
Indicates all other statements within a statement block

The Source Listing Backtrace displays the source of the each statement:

- The order of the statements reflect the program's execution path.
- The top statement is the first statement executed; the bottom statement is the most recently executed statement.

Navigate Through Program Execution

You can trace your program's execution path by using the following PF keys:

- Using PF5, PF9, and PF11 to trace execution backward

- Using PF6, PF10, and PF12 to trace execution forward
- Setting the Source Listing Profile Stepping Amount, and then using PF9 and PF10 to automatically trace the execution backward or forward

End a Source Listing Backtrace Session

1. To turn off the Source Listing Backtrace display mode, press Clear or PF3. CA InterTest for CICS redisplay the Source Listing Breakpoint screen positioned at the last breakpoint.
2. To toggle between the Source Listing Backtrace and the Backtrace Summary, press PF2.

Set the Code Counting Coverage Option

CA InterTest for CICS has many options for specialized testing needs. The code coverage option lets you see the number of times an Assembler statement was executed.

The initial setting for Code Counting is OFF, which is the default. However, you can use the Source Listing Profile screen to turn it ON or OFF again at any time after you have selected monitoring for a program.

How the counts are displayed or not displayed is handled by the primary line command COUNTS and its associated parameters.

This feature does cause overhead during program monitoring and should be turned off as soon as it is no longer needed.

To change the setting for Code Coverage, perform the following steps:

1. Type PROFILE on the command line and press Enter, or press PF4 to access the Profile screen from any Source Listing Screen. The Code Counting= field shows the current setting.
2. Overtyping the current value with one of the following:
 - **YES**
Enables code counting and turns on the COUNTER display feature.
 - **NO**
Removes the COUNTER display and stops the counting feature.
3. Press Enter to process the new setting(s) and to return to the Source Listing Display. The following screen shows the Code Coverage feature.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= PL1DEMO  Option #      Stmt #      Search=      Margin= 01
-----
----- TASKNUM | ?00000.
-----+-----
- 587.1          | 0 F0 F0 F6 F0 F0 F0 F0 'X */,NOPROB); | COUNTER |--
- 588.1          |          END; | 0000001 |
- 589.1          |          /* EXEC CICS DELETEQ TS QUEUE(TSQ_N |
- 590.1          |          DO; | 0000001 |
- 591.1          |          DCL DFHENTRY_BF6577CC_AC4F3B6B BASE | I
- 592.1          | ER ASSEMBLER) ENTRY(*,CHAR(8)); |
- 593.1          | CALL DFHENTRY_BF6577CC_AC4F3B6B(' | 0000001 | 0
    
```

```

- 594.1      80 00 03 00 00 21 00 F0 F0 F6 F1 F0 F0 F0 F0 | 00000001 | N
- 595.1      1 2          END;                          | 00000001 |
- 596.1      1 1          GOTO CONTINUE_TASK;            | 00000000 |
- 597.1      1 1          END;                          | 00000000 |
- 598.1
- 599.1      1          GOTO SENDSCR1;                   | 00000000 |
- 600.1
- 601.1      1          WRITETSQ:                         | 00000001 |
- 602.1          TS_SWITCH = ' ';

```

Indirect Commands

The CA InterTest Indirect Commands facility lets you specify a set of indirect commands to be executed in one of two ways:

- Automatically, when a breakpoint pointing to that set of commands is reached
- On demand, when you are at a breakpoint and use the Breakpoint Primary Option menu to resume execution with an indirect command (the fastpath is =4.5 from any breakpoint)

By using indirect commands, you can dynamically modify or correct programs without having to leave your test session to recompile your program.

With the indirect commands facility, you can do the following actions:

- Change the flow of control in your program
- Test conditions based on specified variables
- Change the value of specified variables
- Create and execute commands as a group, like adding a new subroutine
- Automatically resume execution of your program at the same or different location
- Define abbreviations for variables with long names

Code Indirect Commands

CA InterTest for CICS provides an Indirect Commands facility for coding the statements. You can access it from the Source Listing or from the ITST menus.

To access the Indirect Commands facility from the Source Listing facility, type ICMDS in the command line or 11 in the Option # field and press Enter.



Note: Tab to the More: + field and press Enter to view labels of Options 6 to 15. Notice Option 11 is Indirect commands, and Option 12 is Breakpoint options.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==> ICMDS
Program= PL1DEM0 Option #      Stmt #      Search=      Margin= 01
-----
----- TASKNUM | ?00000.

```

```
-----+-----
- 636.1          CONTINUE TASK:
- 637.1          MAPNUM = '00';
A ==>.1      1    TASKNUM = TASKNUM + 1;
.
.
.
```

To access the Indirect Commands facility from the menus, access the Program Monitoring menu (fastpath is ITST 2.1), and select the Commands option. Detailed steps follow.

1. Select option 2 Monitoring on the Primary Option Menu. The Monitoring Menu appears.
2. Select option 1 Programs. The Program Monitoring screen appears.
3. In the Program field, type the name of the program for which you want to create Indirect Commands.
4. Type S to the left of the Command option (Indirect commands). Press Enter.

```
----- CA InterTest for CICS PROGRAM MONITORING -----
-----
COMMAND ==>

Type information and S to set or R to remove option(s) below.

Program . . PL1DEMO_      Program name (or .ALL, .OPTIONS or generic)
User ID . . _____    User (or .ANY) for whom the program is monitored

Option      Description                                          More:
+
- Status    Display and/or remove monitoring options (S only)
- Monitor   Monitoring (R removes monitoring and all options previously set)
- UBP       Unconditional breakpoints (specific program only)
- CBP       Conditional breakpoints (specific program only)
- RBP       Breakpoints for CICS, DB2, DL/I or external CALL requests
- Stmt Trace Statement tracing and data monitoring (COBOL only)
- New copy  Fetch new copy of program and reset monitoring options (S only)
s Commands  Indirect commands defined for a specific COBOL or PL/I program
- Replace   CICS resource name replacement options
- Protect   Storage protection monitoring options
- Special   Other options (storage allocation, file updating, etc.)
- Composite Monitor multi-CSECT program's separately compiled components

PF1 Help    2          3 End      4 Return   5          6
PF7 Backward 8 Forward  9          10         11         12
```

When you choose Commands from the Program Monitoring menu, another screen lets you set the terminal ID (terminal ID, .ANY or .NO) where the indirect commands take effect, and the user ID who executes the program (.ANY or a specific ID).

Normally you can accept the defaults. However, these entries need to match any existing monitoring entry for the program, which can be viewed on the Monitoring Status display.

```
CA InterTest MONITORING COMMAND BUILDER - INDIRECT COMMANDS      24

SET option to EDIT indirect commands defined for execution in:
                      PROG=PL1DEMO

Term ID (or .ANY or .NO) where commands will take effect:      ----
User ID (or .ANY) who will execute the program:                 -----

PF1 Help    2          3 End      4 Return   5          6
PF7         8          9          10         11         12
```

5. Press Enter to display the Indirect Commands facility.

The Indirect Commands Facility

Either access method brings you to the Indirect Commands facility shown in the following screen. This example shows the indirect commands that can be used to fix the ASRA condition in our PL1DEMO demo program, without recompiling. To do so, follow these steps:

1. Type the statement numbers and the text of the indirect commands on the Indirect Commands screen, as shown in the following screen. For details, see the Indirect Commands section in [Breakpoint Activities \(https://docops.ca.com/display/CAITSD11/Breakpoint+Activities\)](https://docops.ca.com/display/CAITSD11/Breakpoint+Activities) and online Help .
2. After coding the statements, press PF3 to exit the Indirect Commands facility. If you accessed the facility from the ITST menus, use =X to exit the menus.

```

CA InterTest MONITOR COMMAND BUILDER -  PL1DEMO  INDIRECT COMMANDS      24
      Term ID (or .ANY or .NO): G001                                Delete ALL: NO

      LINE  COMMAND
      -----
00010 /* INITIALIZE TASKNUM */
00020 MOVE 0 TO TASKNUM
00030 EXIT

      1 Help      2          3 End      4          5          6
      7 Backward  8 Forward  9 Top     10 Bottom  11         12
CAIN1261 Indirect command(s) added.

```

Control the Flow of Execution

When running your program, if a breakpoint points to indirect commands, the commands are automatically executed without stopping first. Execution is returned to the program according to the exit command that you specified in your indirect commands.

To return execution to your program following indirect command processing, use one of the following exit commands:

- **BREAK**
CA InterTest for CICS halts indirect command processing and issues a breakpoint display from where the indirect commands were invoked in the program.
- **GOTO**
CA InterTest for CICS resumes execution at the program statement number, offset, paragraph name, or indirect command statement number specified after the GOTO command.
- **EXIT**
CA InterTest for CICS resumes program execution at the breakpoint location from which the indirect commands were invoked and continues debugging the program until the next breakpoint.
- **RUN**
CA InterTest for CICS resumes program execution at the breakpoint location from which the indirect commands were invoked and ignores all subsequent breakpoints.

Attach the Indirect Command to Breakpoints

To have the indirect commands automatically executed by your program, set an unconditional, conditional, or request breakpoint that points to the first indirect command statement number. When you set a breakpoint with this option, each time the breakpoint takes effect the indirect commands execute.

From the Source Listing facility, set the unconditional or conditional breakpoint as follows:

1. Display the line in your Source Code where you want to set the breakpoint to execute the indirect commands.
2. Type BPO in the command line or 12 in the Option # field (Breakpoint Options), tab down to the line where you want to set the breakpoint, type U for unconditional or C for conditional, and press Enter. The Breakpoint Locations menu appears, letting you set a number of breakpoint options.
3. Tab to the following option near the bottom of the panel:
Statement no. of indirect command(s) to be executed: _____
4. Type the first indirect command statement number (leading zeroes are optional), as shown in the following screen.

```
CA InterTest MONITORING COMMAND BUILDER - PL/I BREAKPOINT LOCATIONS      11
```

```
SET breakpoint options for  PROG=PL1DEMO  for location:
```

```
        635.1      1      NOPROB:
```

```

All Procedure Names:  _
Enter 'n' to stop only every n'th time:
Term ID (or .ANY or .NO) where breakpoints will take effect:  U057
Term ID (or .ANY) that will receive the breakpoints:         U057
Statement no. of indirect command(s) to be executed:         10__
User ID (or .ANY) who will execute the program:               .ANY
After:  _
```

```
PF1 Help      2          3 End      4 Return    5          6
PF7           8          9          10         11         12
```

5. Press Enter to process this menu.
6. If you are setting a conditional breakpoint, type the condition on the Conditional Breakpoint screen and press Enter.
Your Source Listing display shows the indirect command to execute on the line above the statement with the U or C indicator.
7. To set a request breakpoint that executes an indirect command, follow the usual steps to set the request breakpoint from the menus. On the Request Breakpoint Selection menu, also type the statement number of the indirect command to be executed, as shown in the following screen.

CA InterTest MONITORING COMMAND BUILDER - REQUEST BREAKPOINT SELECTION 13

Set one or more types of Request Breakpoints in:
PROG=PL1DEMO

```

_ ALL commands          _ DL/I    _ DB2    _ CALLs
- Address, Assign,     - Storage Control    - BMS
  Handles, Push, Pop   - Program Control    - Trace Control
_ Terminal Control     - Interval Control   - Dump Control
x File Control         - Task Control       - Batch Data Interchange
_ TD Control           - Journal Control    - Built-In Functions
_ TS Control           - Syncpoints         - Sys Prog Functions
_ Web access           - Business Trans     - 3270 Bridge

Enter 'n' to stop only every n'th time:
Term ID (or .ANY or .NO) where breakpoints will take effect:  ----
Term ID (or .ANY) that will receive the breakpoints:          ----
Statement no. of indirect command(s) to be executed:          100__
User ID (or .ANY) who will execute the program:               -----

PF1 Help      2          3 End      4 Return   5          6
PF7           8          9         10         11         12
    
```

Identify Indirect Commands on the Monitoring Status Display

The following monitoring status display shows the entries for an unconditional breakpoint set at statement #635 that will execute indirect command statement number 10. The indirect commands are listed as a CMD option to the program.

```

----- CA InterTest for CICS  MONITORING STATUS -----
COMMAND ==>

Type + to expand or - collapse option levels displayed below,
or R to remove option(s).

Option  Description              Attributes
- - PL1DEMO Program monitor entry          PL/I
  |      Waiting at breakpoint  Task 00066, ABP since 03:17 p.m.
- - |-.ANY User monitoring options       Active
  |      Symbolic listing file  PROTSYM
- - | -UBP Unconditional breakpoint    #635
  |      Option ID              DB6BDAD0
  |      From, to terminals     U057, U057
  |      Execute indirect command 10
- - | -CMD Indirect commands defined    U057
  |      Option ID              57208DFD
- - | -SLB Source listing breakpoints   U057
  |      *** End of data ***

PF1 Help      2 Refresh   3 End      4 Return   5 Collapse  6 Expand
PF7 Backward  8 Forward  9          10         11         12
    
```

Review, Change or Delete Indirect Commands

From the Monitoring Status display, you can remove any monitoring entry.

1. Type R next to the UBP option and press Enter to remove the breakpoint executing the indirect commands. Type R next to the CMD option to remove the indirect commands.

2. To change, add, delete, or review indirect commands directly from the Source Listing facility, type ICMDS in the command line or 11 in the Option # field.
CA InterTest for CICS displays the indirect commands for the default terminal, which is either the terminal you are working at, or the terminal specified on the Source Listing Profile.

For more information on the Indirect Commands facility, see [Breakpoint Activities \(https://docops.ca.com/display/CAITSD11/Breakpoint+Activities\)](https://docops.ca.com/display/CAITSD11/Breakpoint+Activities).

Replacement, Protection, and Special Options

CA InterTest for CICS has many options for specialized testing needs. A few are explained here. For an explanation of all the monitoring options, see [Monitoring Menu Options \(https://docops.ca.com/display/CAITSD11/Monitoring+Menu+Options\)](https://docops.ca.com/display/CAITSD11/Monitoring+Menu+Options).

You can set or remove options from the Monitoring Command Builder menus. For details, see [Set Options from the Monitoring Menus \(see page \)](#).

Replacement Options

Replacement options let you dynamically change the names of CICS resources such as programs, files, transient data queues, and temporary storage.

One reason to use replacement options is to correct errors so you can continue testing without recompiling. For example, suppose CA InterTest for CICS halts your program at an automatic breakpoint to prevent an AEIL abend caused by an incorrect file name. You can use the Replace File Name option to dynamically change the file name so testing can continue.

Another reason for using replacement options is to use different resources during testing than those hard coded in the program. For example, you can specify that the program use a test file rather than the production file named in the program.

Protection Options

When CA InterTest for CICS monitors a program, it prevents the program from modifying storage areas it does not own and issuing non-CICS instructions.

Protection options let you override the CA InterTest for CICS default rules for modifying main storage, the CSA, and load modules. The protection options, which are password protected, provide extra flexibility for special testing needs. These options let a program to do the following modifications:

- Modify any area of main storage
- Modify areas in the CSA or CWA
- Modify a load module

These options can also prevent a program from modifying specific areas of storage not normally protected.

Special Options

Special options alter the CA InterTest for CICS standard monitoring procedures and help you design customized testing scenarios.

The No File Updating option prevents a monitored program from updating files. This option is in testing because your file data is unchanged at the end of program execution. This means:

- You can repeatedly test a program without restoring your files.
- Programs can share the same file without interfering with each other's work.
- A test program can use a production file without affecting data integrity.

Other options provide additional functions. For example:

- The FOL option lets CA InterTest for CICS continue monitoring even after a program branches directly to another program (wild branch).
- The MXS option limits the amount of CICS storage a program can use.
- The MXR option limits the number of CICS requests a program can issue.

Composite Module Testing

Composite support lets you take advantage of all CA InterTest for CICS functions when you test composite modules. A *composite module* consists of separately compiled modules link-edited in one load module. These modules can be written in different languages and compiled at different times. Composite support lets you test and debug a called subroutine as if it were a separate program -- with full symbolic support.

Example

Suppose a PL/I program has several subroutines, some written in PL/I, some written in Assembler and PL/I. You can test and debug any of the subroutines just as if it were a main program by setting breakpoints, setting monitoring options, displaying the source listing, and displaying and modifying main and auxiliary storage.

A composite module can be defined using one of two methods:

- Execute batch program IN25LINK after the link-edit step for the composite module
- Specify the COMPOSITE command in the Source Listing Display *before monitoring is activated for the program*

When you set monitoring for a composite module defined by the IN25LINK method, CA InterTest for CICS informs you that composite support can be set, as shown in the following message:

```
CAIN4561 Press PF3 to display composite menu for PLICOMP1
```

When you press PF3, CA InterTest for CICS displays the link-edit information for the main program and its subroutines. You can change this information online.

CA InterTest for CICS - Composite Support Builder
 COMMAND ==>
 Define composite support for PL1DEML and press PF5.
 CAIN2620 Not all subprograms are selected

SCROLL: PAGE
 Row 00001 of 00006

* Link name	* Monitor	* Offset	* Length	* Language	* Comments
S PL1DEML1	PL1DEML	80	248	PL/I 4.3	
S PSBIN251	PSBIN25	EE8	1B0	PL/I 4.3	
S ASBIN25	ASBIN25	1098	17C	HLASM 6.0	
- @PL1DEML	-----	1218	10		Symbolics NOT available
- PLIXOPT	-----	1228	8		Symbolics NOT available
- @PSBIN25	-----	1230	8		Symbolics NOT available

PF1 Help 2 3 End 4 5 Process 6 RFind
 PF7 Backward 8 Forward 9 10 11 12

Press PF5 to set composite support so that each program can be tested separately.

Dump Analysis with CA SymDump

CA SymDump for CICS, the CA InterTest for CICS optional symbolic dump facility, is a powerful online tool that automatically pinpoints the source statement responsible for an abend. CA SymDump for CICS was designed to complement CA InterTest, especially in production regions where programs are not usually monitored. Working with CA InterTest, CA SymDump for CICS lets you bring a dump back to life. Familiar CA InterTest for CICS Source Listing screens and other facilities, such as the ability to view and modify main and auxiliary storage, make it easy to resolve application dumps.

CA SymDump for CICS is option 5 Dump analysis on the CA InterTest for CICS Primary Option Menu.

```
----- CA SymDump for CICS PRIMARY OPTION MENU -----
OPTION ==>

1 Analysis            - Display/select captured CICS dumps/traces
2 Tracing            - Capture CICS internal trace for analysis
3 Configuration      - Display/modify CA SymDump initialization parameters
4 Start               - Start dump capture facility
5 Stop                - Stop dump capture facility
6 Source             - Display/select program source files/listings
7 Status/Maintenance - Product status and maintenance functions
8 What's new?        - Display information about CA SymDump for CICS
X Exit                - Terminate CA SymDump menu processing
```

CA SymDump for CICS V10.0

Copyright © 2016 CA. All rights reserved.

PF1 Help 2 3 End 4 Return 5 6
 PF7 8 9 10 11 12

With CA SymDump for CICS, you can do the following tasks:

- Analyze dumps symbolically online, using CA InterTest for CICS source code listings
- Resolve CICS production dumps from any region (for example, debug production abends from your test region)

- Manage your dump data set, selectively retaining the dumps you want to view and print, and set configuration parameters to discard ones you do not need and suppress capture of duplicate dumps

When you use CA SymDump for CICS to view a dump, you can select any of the following displays for immediate viewing:

- **Formatted Displays**
Intelligently formatted displays of the dump contents, that often pinpoints the reason for the abend. Displays include: analysis, source, abend help, program call trace, registers, last CICS screen, files accessed, and programs, transactions and terminals referenced.
- **Task Areas**
CA InterTest for CICS CORE displays for such areas as the Commarea, the EXEC Interface Block, and the Transaction Work Area.
- **Language Areas**
CA InterTest for CICS CORE displays for such areas as COBOL working storage and COBOL BLL cells.
- **Database Areas**
CA InterTest for CICS CORE displays for DB2 and DL/I.
- **CICS System Areas**
CA InterTest for CICS CORE displays for such areas as the Common System Area and the Common Work Area.

Example

Suppose a program not being monitored by CA InterTest for CICS abends with an ASRA. With CA SymDump, you can select Abend analysis display and frequently identify the problem immediately. In the following screen, you can see that abend analysis gives abending instruction and operands -- even without symbolics.

```
-----CA SymDump for CICS ABEND ANALYSIS -----
COMMAND ==>

Type S to display main storage at the data location address.

Prog: PL1DEMO      Code: ASRA      Tran: DEMP      Appl: A01IC9NL      Date: 11/28/2000
User: CICSUSER    Task: 00094      Term: G001     Node: A55TG001     Time: 12:04:06

Data Type          Location  Value
Abended by                Operating system
Abend type                Program check, data exception (0C7)
Abend in CSECT            PL1DEM01 + 00000536
AMODE, EXECKEY            31, USER
PSW at time of abend      078D0E00 8C1068EC 00060007 0B99A000
_ Abending instruction    0C1068E6  FA20D5DD3334
                           AP      X'5DD'(3,R13),X'334'(1,R3)
_ Operand 1 storage        0BA0A46D  000000
                           Invalid packed decimal data
_ Operand 2 storage        0C108A1C  1C
                           Packed decimal data

PF1 Help      2          3 End          4 Return      5          6
PF7 Backward  8 Forward    9          10           11          12
```

The Abend Analysis identifies exactly why and where the abend occurred. It gives the abending instruction and storage values for each operand of the instruction. Abend analysis information is available whether or not you have symbolics available for the abending program.

In the previous example, the description of Operand 1 tells you its storage value is invalid for its data type. It should be a packed decimal. Using this display only, you know what triggered the ASRA!

If you get symbolics for the abending program (either before or after the abend) you can select the Source display and see the breakpoint at the statement that triggered the abend. This is available even though CA InterTest for CICS was not active in the region when the abend occurred.

```
CA SymDump for CICS - PROTSYM FILE SOURCE LISTING DUMP ANALYSIS
COMMAND ==>
Program= PL1DEMO  Option #      Stmt #      Search=      Margin= 01
-----+-----
-          CONTINUE_TASK:
-          MAPNUM = '00';
A ==> 1 0    TASKNUM = TASKNUM + 1;
==>
==> ABEND/DUMP CODE: ASRA
==>
==>      Press PF1 for a detailed description.
==>
- 96 1 0    IF TASKNUM > 2 THEN
-          GOTO END_SESSION;
- 97 1 0    IF TASKNUM = 1 THEN
-          GOTO SENDSCR3;
- 98 1 0    GOTO END_SESSION;
-
- 99 1 0    SENDSCR3:          /** ASRA CONGRATULATION SC
-          /* EXEC CICS SEND MAP('DASRA')
-                               MAPSET('IN25PMP')
-                               WAIT
-                               ERASE */
-
```

Now you can use CA InterTest for CICS features to solve the problem. In this example, you already know the contents of TASKNUM are invalid from the Abend Analysis display. However, if you need to, you can display the contents of any data item. You might want use the backtrace facility to trace the program's execution path, or you might want to inspect the file. CA InterTest for CICS provides all the tools you need to diagnose and resolve the dump.

PL/I Advanced Demo Session

The advanced demo session demonstrates solutions to several situational scenarios: limiting CICS requests, detecting storage violations, debugging composite modules. If you are a new user, you might want to postpone this advanced demo session until you are more comfortable with the basic features of CA InterTest for CICS.

- [Demo Preliminaries \(see page 240\)](#)
- [Execute the Demo Program \(see page 246\)](#)
- [Option 01 Replace a File Control ID \(see page 247\)](#)
- [Option 02 Limit CICS Storage and Requests \(see page 251\)](#)
- [Option 03 Prevent a Program from Updating a File \(see page 255\)](#)
- [Option 04 How to Detect a Storage Violation \(see page 261\)](#)

- [Option 05 How to Test a Composite Module \(see page 264\)](#)



Note: Only a handful of CA InterTest for CICS features are demonstrated in this demo session. For more information, see [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging) or the Help facility.

This article consists of the following independent sections, each detailing a different advanced CA InterTest for CICS feature:

- **Option 01**
Replace a file name.
- **Option 02**
Limit the number of CICS requests issued by a program.
Limit the amount of storage obtained by a program.
- **Option 03**
Prevent a program from updating a file.
Use request breakpoints and the FILE facility.
- **Option 04**
Detect and prevent a storage violation.
- **Option 05**
Test a composite module.

Before proceeding with the demo session, you must complete the steps outlined in the following section.

Demo Preliminaries

The advanced demo session uses PL1DEMO just as the basic session did. Before beginning, you need to set some breakpoints that will be used in different sections of the advanced demo session. The three breakpoints you need to set for PL1DEMO are shown on the following Monitoring Status screen.

```
----- CA InterTest for CICS  MONITORING STATUS  -----
COMMAND ==>

Type + to expand or - collapse option levels displayed below,
or R to remove option(s).

Option  Description  Attributes
- - PL1DEMO  Program monitor entry  PL/I
- - |-.ANY  User monitoring options  Active
- - |  Symbolic listing file  PROTSYM
- - | -UBP  Unconditional breakpoint  #1308
- - |  Option ID  99B13F53
- - |  From, to terminals  U057, U057
- - | -RBP  Request breakpoint(s)  REWRITE
- - |  Option ID  8E495266
- - |  From, to terminals  U057, U057
- - | -UBP  Unconditional breakpoint  'LOOPRTN'
- - |  Option ID  7D41117C
- - |  From, to terminals  U057, U057
```

```

- | -SLB Source listing breakpoints U057
*** End of data ***

PF1 Help      2 Refresh    3 End        4 Return     5 Collapse   6 Expand
PF7 Backward  8 Forward    9           10          11          12
    
```

The details of setting these breakpoints are provided in the next sections.

Set Unconditional Breakpoints

To begin, you will need to bring up the Source Listing display for the PL/I demo program. This was covered in the basic demo session .

1. Press **PF4** to display the Profile panel. Verify that the Display window is set to N and that the AutoKeep Display is ON. If not, change either or both of them and press Enter.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING PROFILE
COMMAND ==>>
Program= PL1DEMO Option # Stmt # Search= Margin= 01
OPTS 1 Data xref 2 Aggregate 3 Stor reqs 4 Stat stor 5 Var stor More: +
      6 Offsets 7 Gend code 8 Err msgs 9 Srch fwd 10 Srch bwd
PFKS 1 Help 2 3 End 4 Auto prms 5 Monitor 6 Menu
      7 Backward 8 Forward 9 Next Wnd 10 11 12 Status
-----+-----
Display window = N N (None), T (Titles), R (Registers),
                K (Keep), P (Program)
PF7/8 amount = PAGE PAGE, HALF, STOP, or a number from 1 to 9999
Step Timing = BEFORE Stop Before or After the next STMTS is executed
Stepping amount = 001 The number of STMTS to execute
Auto-stepping = OFF ON to activate; press PF4 to change values
Source List BKPT = ON OFF to use the detailed breakpoint display
From terminal ID = U006 Terminal ID where the program will execute
BKPT terminal ID = U006 Terminal ID to receive the breakpoint displays
User ID = .ANY User ID who will execute this program
AutoKeep Display = ON OFF to deactivate
Code Counting = OFF ON to activate Code Coverage
SDF = DATA DATA for Structure Display Format
    
```

2. Press **PF3** to return to the listing.
Now you will set unconditional breakpoints at three procedure names. Setting unconditional breakpoints lets you halt the program so that you can set options and check the contents of data items.
3. Start by searching for the label READATA. Tab to the Search= field, type **I readata** in the command line and press Enter.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>> l readata
Program= PL1DEMO Option # Stmt # Search= Margin= 01
-----+-----
- 18.1 /* INTERTEST PL/I DEMO PROGRAM 10/01/89 3.1.200
- 19.1 PL1DEMO: PROC(DFHEIPTR,COMMAREA_PTR) OPTIONS(MAIN REENT
- 20.1 1 DCL 1 DFHCNSTS STATIC, /* CONSTANTS USED BY TRANSLATOR
- 21.1 2 DFHLDVER CHAR(22) INIT('LD TABLE DFHEITAB 530.
- 22.1 2 DFHEIB0 FIXED BIN(15) INIT(0),
- 23.1 2 DFHEID0 FIXED DEC(7) INIT(0),
- 24.1 2 DFHEICB CHAR(8) INIT(' ');
- 25.1 1 DCL DFHEPI ENTRY, DFHEIPTR PTR;
- 26.1 1 DCL 1 DFHEIBLK BASED (DFHEIPTR),
- 27.1 02 EIBTIME FIXED DEC(7),
- 28.1 02 EIBDATE FIXED DEC(7),
- 29.1 02 EIBTRNID CHAR(4),
- 30.1 02 EIBTASKN FIXED DEC(7),
- 31.1 02 EIBTRMID CHAR(4),
    
```

```

-   32.1          02   EIBFIL01 FIXED BIN(15),
-   33.1          02   EIBCPOSN FIXED BIN(15),
-   34.1          02   EIBCALEN FIXED BIN(15),
-   35.1          02   EIBAIID  CHAR(1),
-   36.1          02   EIBFNF   CHAR(2),

```

The READATA label appears.

You will set unconditional breakpoints at the statement:

```
GOTO SENDMP00
```

4. Tab to the GOTO SENDMP00 statement.

5. Type **U** to the left of the statement, and press Enter. The U sets a breakpoint that stops *before* the statement executes. You could also type a) to set a breakpoint that stops *after* the statement executes.

```

CA InterTest for CICS  - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>>
Program= PL1DEMO  Option #          Stmt #          Margin= 01
                        Search=
-----+-----
u 1308.1    1          GOTO SENDMP00;
- 1309.1
- 1310.1    1          READATA:
- 1311.1          /* EXEC CICS HANDLE CONDITION  DSIDERR
- 1312.1          ERROR      (GENERR)
- 1313.1          NOTOPEN   (NOPEN)  *
- 1314.1          DO;
.
.
.

```

CA InterTest for CICS sets the breakpoint and redisplay the screen with an uppercase U.

Set an Unconditional Breakpoint for a Label

Now you will set a second unconditional breakpoint at the LOOPRTN label from the Labels Cross-Reference section. When you set the breakpoint from the Labels Cross-Reference section, it is set at the label name and not at the statement number. This is a good method to use if you want to use the breakpoint after recompiling -- the breakpoint is transferred to the new program by label-name, and not by statement-number (which might change between compiles).

1. Type **L.LX** in the command line or type 14 in the Option # field, and press Enter.

The Labels Cross-Reference section of PL1DEMO appears.

```

CA InterTest for CICS  - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>>
Program= PL1DEMO  Option #          Stmt #          Margin= 01
                        Search=
-----+-----
-          LABEL ATTRIBUTE/XREF TABLE
-   LINE.FILE IDENTIFIER          ATTRIBUTES
-   1646.1   CALLCOMP             CONSTANT LABEL
-                                     Refs: 1643.1
-   857.1    CONTINUE             CONSTANT LABEL
-                                     Refs: 846.1
-   635.1    CONTINUE_TASK        CONSTANT LABEL
-                                     Refs: 596.1
-   1193.1   DSIDERR              CONSTANT LABEL
-                                     Refs: 558.1 1269.1
-   704.1    END_SESSION          CONSTANT LABEL
-                                     Refs: 542.1 576.1 640.1 643+
-                                       666.1 859.1 868.1 882+

```

```

944.1 969.1 994.1 101+
1044.1 1068.1 1091.1
1114.1 1137.1 1160.1
1191.1 1391.1 1487.1
1644.1
_      744.1   EXPANDEM          CONSTANT LABEL

```

2. Press PF8 to page forward.
3. Type **U** to the left of the LOOPRTN label, and press Enter. The U sets a breakpoint that stops *before* the label name. You could also type a) to set a breakpoint that stops *after* the label name.

```

CA InterTest for CICS - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ==>
Program= PL1DEMO Option # 9 Stmt # Margin= 01
Search= LOOPRTN
-----+-----
      ATTRIBUTE/XREF TABLE-
-      LINE.FILE IDENTIFIER          ATTRIBUTES
      515.1   LENGTH                  Refs: 709.1 1213.1 1235.1 1+
u      1435.1  LOOPRTN                 BUILTIN
      773.1   MAPNUM                   CONSTANT LABEL
      773.1   MAPNUM                   Refs: 970.1 995.1
-      773.1   MAPNUM                   AUTOMATIC CHARACTER(2) UNAL+
      773.1   MAPNUM                   Refs: 906.1
      773.1   MAPNUM                   Sets: 635.1 860.1 922.1 947+
      773.1   MAPNUM                   972.1 997.1 1022.1 16+
-      1536.1  MORE_ENTRY               CONSTANT LABEL
      1536.1  MORE_ENTRY               Refs: 903.1
.
..

```

CA InterTest for CICS sets the breakpoint at the LOOPRTN label and redisplay the screen with an uppercase U.

How to Set Request Breakpoints

You can set request breakpoints to halt a program before CICS commands and macros and other calls, such as calls to DL/I or DB2. You can set request breakpoints prior to all CICS commands (as you might have done with IBM's CEDF) or prior to specific commands, such as File Control or Program Control commands. Request breakpoints make it easy for you to halt your program at various points so you can check program processing. For example, you can halt the program before every READ command, single-step through the read instructions, and then check main storage to make sure the program has read the correct record.

For option 03 of the advanced demo session, you need to set request breakpoints at all File Control REWRITE commands in the demo program. To do this, access the Request Breakpoint Selection menu.

1. Press PF6 to return to the Primary Option Menu.
2. Select 2 Monitoring.
The Monitoring Menu appears.
3. Select 1 Programs.
The Program Monitoring screen appears.

```

----- CA InterTest for CICS PROGRAM MONITORING -----
-----
COMMAND ==>

```

Type information and S to set or R to remove option(s) below.

```

Program . . PL1DEMO_      Program name (or .ALL, .OPTIONS or generic)
User ID . . _____   User (or .ANY) for whom the program is monitored

Option      Description                                     More:  +
- Status    Display and/or remove monitoring options (S only)
- Monitor   Monitoring (R removes monitoring and all options previously set)
- UBP       Unconditional breakpoints (specific program only)
- CBP       Conditional breakpoints (specific program only)
s RBP       Breakpoints for CICS, DB2, DL/I or external CALL requests
- Stmt Trace Statement tracing and data monitoring (COBOL only)
- New copy  Fetch new copy of program and reset monitoring options (S only)
- Commands  Indirect commands defined for a specific COBOL or PL/I program
- Replace   CICS resource name replacement options
- Protect   Storage protection monitoring options
- Special   Other options (storage allocation, file updating, etc.)
- Composite Monitor multi-CSECT program's separately compiled components

PF1 Help    2          3 End      4 Return   5          6
PF7 Backward 8 Forward  9         10        11        12
    
```

4. In the Program field, type the name of the program for which you want to set a request breakpoint.
5. Type S to the left of RBP (request breakpoints) and press Enter.
The Request Breakpoint Selection menu appears on which you specify the type of request breakpoint, such as all CICS commands, File Control, Task Control, and so on.
6. Tab to the File Control field and type S.
7. Press Enter.

CA InterTest MONITORING COMMAND BUILDER - REQUEST BREAKPOINT SELECTION 13

Set one or more types of Request Breakpoints in:
PROG=PL1DEMO

```

_ ALL commands          _ DL/I          _ CALLs
- Address, Assign,     - Storage Control - BMS
  Handles, Push, Pop   - Program Control - Trace Control
- Terminal Control     - Interval Control - Dump Control
s File Control         - Task Control    - Batch Data Interchange
- TD Control          - Journal Control - Built-In Functions
- TS Control          - Syncpoints     - Sys Prog Functions
- Web access         - Business Trans - 3270 Bridge
- Channel/Container

Enter 'n' to stop only every n'th time:
Term ID (or .ANY or .NO) where breakpoints will take effect:
Term ID (or .ANY) that will receive the breakpoints:
Statement no. of indirect command(s) to be executed:
User ID (or .ANY) who will execute the program:

PF1 Help    2          3 End      4 Return   5          6
PF7         8          9         10        11        12
    
```

The File Control Request Breakpoint Selection menu appears.

8. On this menu you can select the specific File Control commands where you want to set request breakpoints. In this case, select REWRITE to halt the program prior to each File Control REWRITE command.

9. Tab to the REWRITE command field.

10. Type an **s** and press Enter.

```
CA InterTest for CICS MONITORING COMMAND BUILDER - REQUEST BREAKPOINT SELECTION
```

```
Set one or more types of File Control COMMANDS IN:
      PROG=PL1DEMO
```

```
_ All commands
```

```
  _ READ
  _ WRITE
  s REWRITE
  _ DELETE
```

```
 .
 .
 .
```

CA InterTest for CICS sets request breakpoints prior to each File Control REWRITE command in the demo program and redisplay the Program Monitoring Menu.

The next topic is necessary only if you want to use the composite support option. Composite support is a CA InterTest for CICS feature designed to aid programmers who are responsible for testing and debugging called subroutines.

- If you do not want to use the composite support option, skip to [Execute the Demo Program \(see page 246\)](#).
- If you do want to use the composite support option, perform the steps in Composite Support that follows.

Composite Support

Composite support lets you use CA InterTest for CICS to test and debug composite modules. A *composite module* is a load module (defined in the PPT) that consists of separately compiled or assembled parts brought together when the module is link-edited. In the demo session, the part of the composite module that receives control from CICS is referred to as the *main program*; the remaining programs are referred to as *subroutines*. The main program and subroutines can be written in the same or different languages.

Composite support lets you test and debug a subroutine, such as an external procedure, as if it were a separate program with full symbolic support. This means you can set breakpoints and other monitoring options individually for any subroutine. CA InterTest for CICS also generates automatic breakpoints in the subroutine when it detects an error that would otherwise cause the program toabend.

To demonstrate this feature, you will set composite support for the composite module PL1DEML, to which PL1DEMO links. PL1DEML has two called subroutines that you want CA InterTest for CICS to monitor -- subroutine PSBIN25 written in PL/I, and subroutine ASBIN25 written in Assembler.

Follow these steps:

1. In the Program Monitoring Menu, type =X in the Command line, and press Enter to exit the menus and return to the Source Listing Display.
2. In the Program field, overwrite PL1DEMO with PL1DEML, and press Enter.

From the Welcome Screen you can access the Options Menu. Each of the options is detailed in the sections that follow

3. Press PF2 to go to the Options Menu of the demo.
 The demo program displays the Options Menu, as shown next.

```
*****
****                                     ****
****           CA InterTest Demo Session           ****
****                                 Options Menu             ****
****                                     ****
*****
```

```
01 Replace file control ID
02 Limit CICS storage and requests
03 No file updating
04 Storage violation detection
05 Composite support
```

Key in request: 01

Press ENTER to continue or CLEAR to terminate

Option 01 Replace a File Control ID

This section of the demo session details how to replace a File Control ID. The demo program PL1DEMO is attempting to read and display a record, but the file name specified in the program is incorrect, possibly because it was misspelled.



Important! Before proceeding, be sure you have completed the steps outlined in the [Demo Preliminaries \(see page \)](#) section , and that you have the CA InterTest for CICS Demo Session Options Menu displayed on your screen.

1. Select option 01 and press Enter.
 The demo program displays the following screen, which describes what occurs in this part of the demo session.

```
*****
****                                     ****
****           CA InterTest Demo Session           ****
****      Replace File Control ID Option             ****
****                                     ****
*****
```

The program attempts to read file PROTH. Because the file name has been incorrectly specified, this would cause a DSIDERR condition which would result in an AEIL abend. Instead, here is what will happen:

1. CA InterTest halts the program at an automatic breakpoint.
2. You set the Replace File Control option to change the file name without recompiling.
3. You re-execute the program from the point at which it first tried to read the file.

When this section of the demo completes, you will return to the Options Menu.

Press ENTER to continue or CLEAR to terminate

2. Press Enter.

CA InterTest for CICS halts the demo program at an automatic breakpoint.

Prevent an AEIL Abend

The following screen shows that CA InterTest for CICS halted the demo program at an automatic breakpoint (note the A in column 1) before an AEIL abend could occur.

```
CA InterTest for CICS - PROTSYM FILE  ABEND DETECTED BREAKPOINT
COMMAND ==>>
Program= PL1DEMO  Option #          Stmt #          Search=          Margin= 01
-----+-----
- 1321.1      1          RECKEY = '0000000000';
- 1322.1      1          /* EXEC CICS STARTBR DATASET('PROTH') RIDFLD(RECK
A ==>.1      1          D0;          ==>
==> CICS abend intercepted. Abend code = AEIL. EXEC CICS request was
==> STARTBR. Now stopped AFTER the CALL.
==>
==> Press PF1 for a detailed description.
==>

- 1324.1      1  1          DCL DFHENTRY_BF6577CC_B2C81A6B BASED(ADDR(DFHEI0
- 1325.1      1          ASSEMBLER) ENTRY(*,CHAR(8),*,FIXED BIN(15),*);
- 1326.1      1  1          CALL DFHENTRY_BF6577CC_B2C81A6B(' .^ . . μ.0433000
- 1327.1      1          00 03 00 20 A0 00 F0 F4 F3 F3 F0 F0 F0 F1 'X */,'PROTH'
- 1328.1      1          FHEIB0,RECKEY);
- 1329.1      1  1          END;
- 1330.1      1          REC_LEN = 0;
- 1331.1      1          /* EXEC CICS READNEXT DATASET('PROTH')
- 1332.1      1          SET(REC_PTR)
- 1333.1      1          RIDFLD(RECKEY)
```

What's the problem?

The data set name PROTH is wrong; it should be PROTHLF.

How can it be fixed?

CA InterTest for CICS lets you dynamically correct the file name and continue testing. The following section explains how.

Change the File Name

CA InterTest for CICS lets you dynamically change an incorrect file name in the middle of your test session, allowing you to continue testing without recompiling your program. In this case, you can correct a simple typo in the demo program without interrupting your test session or changing your File Control Table.

To change the file name, you need to access the Program Monitoring Replacement Options menu.

1. Specify RO on the command line.

2. Press Enter.
The Replacement Options menu appears.
The Set Replacement Options menu lets you dynamically change a program name, file name, transient data queue name, or temporary storage ID.
3. Tab to the Replace file name field.
4. Specify the incorrect file name (proth) in the first column and the correct file name (prothlf) in the second column, and press Enter.

CA InterTest for CICS MONITORING COMMAND BUILDER - REPLACEMENT OPTIONS 20

SET one or more options to replace one CICS resource name with another in:
PROG=PL1DEMO

```

Replace program name:          _____ with _____
Replace file name:            proth___ with prothlf_
Replace TD queue name:       _____ with _____
TS selection mask:           _____
TS replacement mask:         _____
    
```

```

Limit monitoring to your terminal - '*' or TERMIID:  _____
User ID (or .ANY) who will execute the program:     _____
    
```

CA InterTest for CICS dynamically replaces the incorrect file name when the program executes. CA InterTest for CICS then redisplay the Program Monitoring menu.

5. Press PF4 until the Source Listing Breakpoint Screen reappears.

Resume Execution

Now that you have dynamically corrected the file name, you are ready to resume program execution. However, you must back up to and execute the statement that contains the EXEC CICS STARTBR command with the incorrect file name, rather than resume execution at the statement that triggered the breakpoint.

1. On the Source Listing Breakpoint screen, tab to the DO statement following the EXEC CICS STARTBR command and type G (go).
2. Press Enter to resume program execution.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>>
Program= PL1DEMO  Option #      Stmt #      Search=      Margin= 01
-----+-----
- 1321.1      1          RECKEY = '0000000000';
- 1322.1      1          /* EXEC CICS STARTBR DATASET('PROTH') RIDFLD(RECK
g ==>.1      1          DO;
- 1324.1      1 1        DCL DFHENTRY_BF6577CC_B2C81A6B BASED(ADDR(DFHEI0)
- 1325.1      1          ASSEMBLER) ENTRY(*,CHAR(8),*,FIXED BIN(15),*);
- 1326.1      1 1        CALL DFHENTRY_BF6577CC_B2C81A6B(' .^ . μ.0433000
- 1327.1      1          00 03 00 20 A0 00 F0 F4 F3 F3 F0 F0 F0 F1 'X */,'PROTH'
- 1328.1      1          FHEIB0,RECKEY);
- 1329.1      1 1        END;
- 1330.1      1          REC_LEN = 0;
- 1331.1      1          /* EXEC CICS READNEXT DATASET('PROTH')
    
```

```

- 1332.1          SET(REC_PTR)
- 1333.1          RIDFLD( RECKEY)
- 1334.1          LENGTH( REC_LEN) */
- 1335.1          1          DO;
- 1336.1          1 1          DCL DFHENTRY BF6577CC B2D0406B BASED(ADDR(DFHEIO)
- 1337.1          ASSEMBLER) ENTRY(*,CHAR(8),POINTER,FIXED BIN(15),*,*,FI
- 1338.1          1 1          CALL DFHENTRY BF6577CC B2D0406B('0000011000001110
- 1339.1          00000001100000001000000001011000000000000111100001111010

```

The program resumes execution and displays the record it has read on your screen as shown in the following screen.
 The demo program has succeeded in reading a record from file PROTHLF (the CA InterTest for CICS Help file). The demo program displays the first record in that file.

```

*****
****
****          CA InterTest Demo Session          ****
****          Replace File Control Identifications ****
****
*****
*****
Record retrieved: * 0000000000    VE0301InterTest ZELP Facil *
                  * ity Master Menu                       *
                  *                                         *
                  *                                         *
Record length = 0960

```

You have now successfully utilized the REPLACE option

Press ENTER to continue or CLEAR to terminate

Review What Happened

In this part of the demo session:

- CA InterTest for CICS halted the demo program at an automatic breakpoint caused by an incorrect file name
- You dynamically corrected the file name and resumed program execution.

Replacing the incorrect file name in the demo program allowed you to continue testing that program without recompiling it. Of course, if one of your programs or the CICS FCT has an incorrect file name, you would eventually have to correct the source code.

In this case the error was caused by a typo. However, the Replacement Options are useful in many situations. For example:

- You can dynamically change a file name to test a program using a different file. This allows you to use a test file without altering the name of the production file hard coded in the program.
- You can change a program name or transient data queue name to test a program or queue other than the ones that are hard coded.

This concludes Option 01 of the advanced demo session. Press Enter to continue with the demo session or press Clear to terminate the session.

Option 02 Limit CICS Storage and Requests

This section of the demo session shows how to limit CICS main storage and CICS requests for a specific program (the demo program). Two CA InterTest for CICS options are detailed in this part of the demo session:

- The MXS option limits the amount of main storage obtained by a task. This option is useful in detecting program errors that cause a task to acquire an excessive amount of storage.
- The MXR option limits the number of CICS requests, commands, and macros that a program can issue. This option is useful in detecting program loops that generate an excessive number of CICS requests.



Important! Before proceeding, be sure you have completed the steps outlined earlier in this article in [Demo Preliminaries \(see page 239\)](#), and that you have the CA InterTest for CICS Demo Session Options Menu displayed on your screen.

1. Select option 02, and press Enter.

The demo program displays a screen that describes what will occur in this part of the demo session.

```
*****
****
****                      CA InterTest Demo Session                      ****
****                      MXR Option - Limit Number of CICS Requests          ****
****                      MXS Option - Limit Amount of CICS Storage           ****
****
*****
```

The program has a loop containing a CICS request and another loop containing a GETMAIN request. Here is what will happen:

1. The program is halted at an unconditional breakpoint before the loops.
2. You set the MXS and MXR options to limit storage and CICS requests.
3. The program continues to execute.
4. CA InterTest halts the program at automatic breakpoints when the limits are exceeded.
5. You remove the options and the program completes execution.

When this section of the demo completes, you will return to the Options Menu.

Press ENTER to continue or CLEAR to terminate

2. Press Enter.

CA InterTest for CICS halts the program at the procedure name LOOPRTN label because you set an unconditional breakpoint there as part of the Demo Preliminaries.

Now you can access Special Options from the Program Monitoring menu to set the options you need for testing.

Set the MXS and MXR Options

1. Type SO in the command line.

- Press Enter.
The Special Options menu appears.
The Special Options menu lets you set several options to change how CA InterTest for CICS monitors a program. You are going to set two options:

- MXS, to limit the amount of CICS main storage acquired by the demo program
- MXR, to limit the total number of CICS requests issued by the demo program

- Specify 50,000 bytes as the maximum amount of main storage the demo program can get, and specify 20 as the maximum number of CICS requests the demo program can issue.

- Tab to the MXS field and type in 135000, as shown next.

- Tab to the MXR field and type in 20, as shown next.

- Press Enter.

```

CA InterTest MONITORING COMMAND BUILDER - SPECIAL OPTIONS                22
      SET one or more options to override the default monitoring rules in:
      PROG=PL1DEMO

      Enter X next to each option desired:

              Source Listing Breakpoint      (SLB)  _
              No file updating                (NUP)  _
              Reentrancy check                (RNT)  _

      Follow monitoring (ON, name, NOPPT) (FOL)  _____
      Number of times to be monitored      (MUS)  _____
      Limit total size of CICS storage      (MXS)  135000
      Limit total number of CICS requests  (MXR)  20
      Structure Display Format (HEX,DATA) (SDF)  _____

      Set local automatic breakpoint ('*', TERMID, .ANY, OFF)  _____
      Limit monitoring to your TERMINAL - '*' or TERMID:      _____
      User ID (or .ANY) who will execute the program:         .ANY

PF1 Help      2          3 End      4 Return   5          6
PF7           8          9          10         11         12
    
```

CA InterTest for CICS limits main storage and CICS requests for the demo program and redispays the Program Monitoring screen.

- Press PF4 until you are returned to the Source Listing Breakpoint screen.

- Press PF5 to continue program execution.
The demo program resumes execution.

Limit CICS Requests

The next screen is an automatic breakpoint display screen. CA InterTest for CICS halted the demo program at an automatic breakpoint because the maximum number of CICS requests (20) was exceeded.



Note: If you are working on a system with LE 370 runtimes, you might reach the automatic breakpoint for the MXS (max storage size) first.

If you do, perform the demo in the following order:

1. Remove the MXS options by typing STATUS, in the command line then 'R' next to the MXS option, and press enter. Resume execution.
2. After stopping for the MXR trigger value, remove it the same as the MXS and continue.
3. The Demo will execute to completion.
The instruction that triggered the automatic breakpoint is within a loop containing a CICS request.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= PL1DEMO  Option #      Stmt #                               Search=          Margin= 01
-----+-----
- 1436.1                DO I = 1 TO 50;
- 1437.1                /* EXEC CICS ASKTIME */
A ==>.1      1 1        DO;          ==>
  ==> MXR (max CICS requests) trigger value exceeded.
  ==>
  ==>   Press PF1 for a detailed description.
  ==>

- 1439.1      1 2        DCL DFHENTRY_BF6577CC_B372BAE6 BASED(ADDR(DFHE
.
.
.
    
```

Remove the MXR Option

Now you can remove the MXR option so that the demo program continues executing. If this were your own program, you would want to examine the instructions that are generating excessive CICS requests.

1. Press PF12, or type STATUS on the command line and press Enter.
The Monitoring Status appears.

```

----- CA InterTest for CICS MONITORING STATUS -----
COMMAND ==>

Type + to expand or - collapse option levels displayed below,
or R to remove option(s).

Option      Description                               Attributes
- - PL1DEMO  Program monitor entry                     PL/I
- - |        Waiting at breakpoint          Task 00087, ABP since 01:02 p.m.
- - |-.ANY   User monitoring options                 Active
- - |        Symbolic listing file         PROTSYM
- - |        -MXR      Maximum CICS requests                     20
- - |        -MXS      Maximum total storage                135000
- - |        -RBP      Request breakpoint(s)             REWRITE
- - |        |                                                DELETE
- - |        |        Option ID                    88A2A671
- - |        |        From, to terminals            U068, U068
- - |        -UBP     Unconditional breakpoint          #1308
- - |        |        Option ID                    813C8815
- - |        |        From, to terminals            U068, U068

More: +
    
```

```

_ - | -UBP      Unconditional breakpoint  'LOOPRTN'
PF1 Help      2 Refresh      3 End          4 Return      5 Collapse   6 Expand
PF7 Backward  8 Forward      9             10           11           12
    
```

2. Remove the MXR option by typing an r next to the MXR breakpoint's entry and pressing Enter. CA InterTest for CICS removes the breakpoint.
3. Press PF3 to return to the Source Listing Breakpoint screen.
4. Press PF5 to resume program execution.

Limit Acquisition of Storage

The next screen is an automatic breakpoint display. CA InterTest for CICS halted the demo program at an automatic breakpoint because the maximum main storage limit of 135,000 was exceeded. In most cases, the instruction that triggered the automatic breakpoint is within a loop that contains a GETMAIN request.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= PL1DEMO  Option #      Stmt #                      Margin= 01
                   Search=
-----+-----
- 1461.1      1 2                CALL DFHENTRY_BF6577CC_B381D92B(' . { . ..δ.0476
- 1462.1      1 2                C0 00 03 00 00 8C 00 F0 F4 F7 F6 F0 F0 F0 F0 'X */ ,GET_
A ==>.1      1 2                END; ==>
  ==> MXS (max storage size) trigger value exceeded.
  ==>
  ==> Press PF1 for a detailed description.
  ==>

- 1464.1      1 1                END;
- 1465.1      1                GOTO SENDMP00;
- 1466.1      1                NOSTORG:
.
.
.
    
```

Remove the MXS Option

Now you can remove the MXS option so that the demo program continues executing. If this were your own program, you would want to examine the instructions that are causing the program to obtain excessive amounts of storage.

1. Press PF12, or type STATUS on the command line and press Enter. The Monitoring Status screen appears.

```

----- CA InterTest for CICS MONITORING STATUS -----
COMMAND ==>

Type + to expand or - collapse option levels displayed below,
or R to remove option(s).

Option      Description                      Attributes
- - PL1DEMO  Program monitor entry            PL/I
  |          Waiting at breakpoint        Task 00087, ABP since 01:13 p.m.
- - |-.ANY   User monitoring options          Active
  |          Symbolic listing file            PROTSYM
r  | -MXS    Maximum total storage           135000
.
.
.
    
```

2. Remove the MXS option by typing an r next to the MXS breakpoint's entry and pressing Enter. CA InterTest for CICS removes the breakpoint.
3. Press PF3 to return to the Source Listing Breakpoint screen.

Demo Program Completes Execution

Press PF5 to resume program execution.

The demo program completes execution. CA InterTest for CICS returns to the demo's Options Menu.

Review What Happened

In this part of the test session you set two options to help test the demo program:

- The MXR option lets you limit the number of CICS requests the program could issue. When the demo program exceeded this limit, CA InterTest for CICS halted the program. This feature helps you find program instructions within a loop that generate an excessive number of CICS requests.
- The MXS option lets you limit the amount of main storage used by the program. When the demo program exceeded the amount of storage you specified, CA InterTest for CICS halted the program. This feature helps you find commands or macros within a loop that are getting excessive amounts of storage.

This concludes Option 02 of the advanced demo session. Press Enter to continue with the demo session or press Clear to terminate the session.

Option 03 Prevent a Program from Updating a File

This section of the demo session details how to execute a program without having it update a file.

The No File Updating option is useful when you are testing a program. Preventing a program from updating a file means your test data remains unchanged at the end of each program execution, so you can test the program repeatedly without having to recreate test data.

Many programs can use the same test file without interfering with each other's work. You can even allow a test program to use a production file because the integrity of the file is preserved.



Important! Before proceeding, be sure you have completed the steps outlined earlier in this article in [Demo Preliminaries \(see page \)](#) and that CA InterTest for CICS Demo Session Options Menu is displayed on your screen. Also be sure that the CA InterTest for CICS checkpoint file (PROTCPF) has been defined to your CICS region and that the file's current status is Open and Enabled.

1. Select option 03, and press Enter.
The demo program displays the following screen that describes what will occur in this part of the demo session.

```
*****
****
****                                CA InterTest Demo Session                                ****
*****
```

```

****                               No File Updating Option                               ****
****                                                                           ****
*****

```

The program reads a record and changes data in the record. Here is what will happen:

1. CA InterTest halts the program at a request breakpoint.
2. You set the No File Updating option to prevent PL1DEMO from updating the file.
3. When PL1DEMO rewrites the file, the file is not updated.
4. After the rewrite you use the FILE facility to confirm that the file has not been updated.

When this section of the demo completes, you will return to the Options Menu.

Press ENTER to continue or CLEAR to terminate.

2. Press Enter.

CA InterTest for CICS displays a request breakpoint (note the R in column 1) shown in the following section's screen.

Halt the Demo Program at a Request Breakpoint

CA InterTest for CICS halted the demo program prior to the first File Control REWRITE command as you specified during the Demo Preliminaries section. The highlighted CALL instruction is part of the generated EXEC CICS REWRITE command. Setting request breakpoints lets you inspect the values of program variables and set options prior to all or specified CICS commands and macros and other program calls.

The following screen shows CA InterTest for CICS halting the program prior to a REWRITE Statement.

```

CA InterTest for CICS - PROTSYM FILE STEP BEFORE BREAKPOINT
COMMAND ==>
Program= PL1DEMO Option # Stmt # Search= Margin= 01
-----
----- CKPTNAME | PROTCPF
----- NUP_AREA | THIS IS NOT A NAME PRIME THE
----- REC_LEN | +00681.
-----
+-----
- 1289.1 | FROM(NUP_AREA)
- 1290.1 | LENGTH(REC_LEN) */
- 1291.1 1 DO;
- 1292.1 1 1 DCL DFHENTRY_BF6577CC_B2AB522B BASED(ADDR(DFHEI0)
- 1293.1 ASSEMBLER) ENTRY(*,CHAR(8),*,FIXED BIN(15));
R ==>.1 1 1 CALL DFHENTRY_BF6577CC_B2AB522B(' \. .. .0423000
- 1295.1 00 03 00 00 40 00 F0 F4 F2 F3 F0 F0 F0 F0 'X */;CKPTNAM
- 1296.1 1 1 LEN); END;
- ==>.1 1 REMBKPT:
.
.
.

```

1. Scroll backward to look at the EXEC CICS REWRITE command.

2. Press PF7.

The following screen appears.

```

CA InterTest for CICS - PROTSYM FILE STEP BEFORE BREAKPOINT
COMMAND ==>

```

```

Program= PL1DEMO  Option #      Stmt #      Search=      Margin= 01
-----
----- CKPTNAME | PROTCPF
----- NUP_AREA  | THIS IS NOT A NAME PRIME THE
----- REC_LEN  | +00681.
-----
+-----
- 1282.1  1  1      CALL DFHENTRY_BF6577CC_B2A39FEB(' 0. .d.0416000
- 1283.1      00 03 00 00 84 00 F0 F4 F1 F6 F0 F0 F0 F0 'X */ ,CKPTNAM
- 1284.1      LEN,NUP_KEY);
- 1285.1  1  1      END;
- 1286.1  1      NUPWRITE:
- 1287.1      NUP_NAME = 'THIS IS NOT A NAME';
- 1288.1      /* EXEC CICS REWRITE DATASET(CKPTNAME)
- 1289.1      FROM(NUP_AREA)
- 1290.1      LENGTH(REC_LEN) */
- 1291.1  1      DO;
- 1292.1  1  1      DCL DFHENTRY_BF6577CC_B2AB522B BASED(ADDR(DFHEI0)
.
.
.

```

According to the EXEC CICS REWRITE command, the demo program updates the file PROTCPF (specified in the DATASET parameter of the EXEC CICS REWRITE command).

 **Important!** If you do not have the AutoKeep Display function running, to view the contents of DATASET(CKPTNAME), you can type Search=ckptname to scroll to where it is defined, and then add it to the Keep window (type K next to the line defining CKPTNAME and press Enter).

The Keep window shows the current value of CKPTNAME is indeed PROTCPF. To return to a breakpoint location, press Clear. However, you are going to set the No File Updating option to prevent the demo program from updating the file. To do so, access the Special Options menu as explained in the following section.

The No File Updating Option

As we did when we set and removed the MXR and MXS options, bring up the Program Monitoring screen. For convenience, use the CA InterTest for CICS fast path feature to navigate to this screen.

1. Type SO on the command line and press Enter. The Special Options menu appears.
2. Tab to the NUP field.
3. Type X, and press Enter.

```

CA InterTest for CICS MONITORING COMMAND BUILDER - SPECIAL OPTIONS      22
      SET one or more options to override the default monitoring rules in:
      PROG=PL1DEMO

      Enter X next to each option desired:

      Source Listing Breakpoint      (SLB) _
      No file updating                (NUP) x
      Reentrancy check                (RNT) _

```

```

Follow monitoring (ON, name, NOPPT) (FOL) -----
Number of times to be monitored (MUS) -----
Limit total size of CICS storage (MXS) -----
Limit total number of CICS requests (MXR) -----

Set local automatic breakpoint (*', TERMID, .ANY, OFF) -----
Limit monitoring to your TERMINAL - '*' or TERMID: -----
User ID (or .ANY) who will execute the program: -----
    
```

```

PF1 Help      2          3 End      4 Return    5          6
PF7           8          9         10         11         12
    
```

CA InterTest for CICS sets this option for the demo program.

4. Type =X in the Command field and press Enter to return to the Source Listing Breakpoint screen.
Before resuming program execution, look at the statement just before the EXEC CICS REWRITE command:

```
NUP_NAME = 'THIS IS NOT A NAME'
```

PL1DEMO is supposed to move the data string THIS IS NOT A NAME to the data field NUP_NAME. If PL1DEMO actually updates the file, this data string displays at the beginning of the first record in that file.

5. Press PF5 to continue program execution.
The demo program resumes execution.

Use FILE to Inspect the Record

CA InterTest for CICS now halts the demo program at an unconditional breakpoint that you set in the Demo Preliminaries section.

```

CA InterTest for CICS - PROTSYM FILE UNCOND BEFORE BREAKPOINT
COMMAND ==>>
Program= PL1DEMO Option #      Stmt #      Search=      Margin= 01
-----
----- CKPTNAME                | PROTCPF
----- NUP_AREA                 | THIS IS NOT A NAME PRIME THE
----- REC_LEN                  | +00681.
-----+-----
- 1305.1          00 03 00 00 50 00 F0 F4 F1 F1 F0 F0 F0 F0 'X */,DFHEID0
- 1306.1          NTLTRAN,NUPOFFCM,NUPONLEN);
U ==>.1          1 1          END;
- 1308.1          1          GOTO SENDMP00;
- 1309.1
- 1310.1          1          READATA:
- 1311.1          /* EXEC CICS HANDLE CONDITION DSIDERR
- 1312.1          ERROR (GENERR)
- 1313.1          NOTOPEN (NOPEN) *
- 1314.1          DO;
.
.
.
    
```

You set the breakpoint at the paragraph immediately following the REWRITE command so you could use the CA InterTest for CICS FILE facility to confirm that the demo program did not, in fact, update the file.

1. Press PF6.
The Breakpoint Primary Option Menu appears.
2. Select option 1 Main Menu.
The Primary Option Menu appears.
3. Select 4 Auxiliary storage.
The Auxiliary Storage Menu appears.
4. Type a 1 in the Option field and protcpf for the file name.

```

----- CA InterTest for CICS  AUXILIARY STORAGE MENU -----
OPTION ==> 1

      Select an auxiliary storage type, specifying optional criteria below.

      1 Files           - Display/select files for access
      2 DB2 database    - Invoke DB2 SQL interface facility
      3 DL/I database   - Access DL/I database
      4 TD queues       - Display/select transient data queues for access
      5 TS queues       - Display/select temporary storage queues for access

      Type specific or generic file/queue name(s):
      (Valid mask characters are * and/or +)

      protcpf_ -----
      .
      .
      .
  
```

5. Press Enter.
The File Selection menu appears.
6. Type S next to the file name and press Enter.
The initial File Facility screen appears.

```

DATATYPE= FC FILEID= PROTCPF  MODE=      LOG=ON  TODEST=      PASSWORD=
FUNC=      SUBFUNC=      RETMETH=      ARGTYP=      SRCHTYP=
MESSAGE=
  RETNRCID=      CHGLEN=
    RCID=
    DATA=      SIZE= 0000
FORMAT= D 00112233 44556677 8899AABB CCDDEEFF *0123456789ABCDEF*
LOC 0000 .....
  
```

```

-----
1 Help      2 Format C  3 End      4 BEGB      5          6 DataType DL
7 Page bwd  8 Page fwd  9 Caps Off 10 Top      11 Bottom   12
  
```

7. Press PF4 to begin a browse of the file.
CA InterTest for CICS displays the first record in the PROTCPF file.

```

DATATYPE= FC FILEID= PROTCPF  MODE=BROWSELOG=ON  TODEST=      PASSWORD=
FUNC= NEXT SUBFUNC=      RETMETH=      ARGTYP=      SRCHTYP=
MESSAGE= CAIN0601 RECORD OBTAINED FOR VIEWING
  RETNRCID=4040404040404040      CHGLEN=
    RCID=
    DATA=      SIZE= 02A9
FORMAT= D 00112233 44556677 8899AABB CCDDEEFF *0123456789ABCDEF*
LOC 0000 .....
    0000   40404040 40404040 4040405C 5C5C4040      ***   DSORG=VSKS
  
```

```

0010  404040D9 C5C3D6D9 C440E3D6 40D7D9C9      RECORD TO PRI  RECFM=FB
0020  D4C540E3 C8C540C3 C8C5C3D2 D7D6C9D5      ME THE CHECKPOIN  LRECL=02A9
0030  E340C6C9 D3C54040 405C5C5C 5C404040      T FILE      ****  BLKSIZE=0000
0040  40404040 40404040 F0F3F1F4 F0F0F0F0      03140000      KEYPOS=0000
0050  00000000 00000000 00000000 00000000      .....      KEYLEN=09
0060  00000000 00000000 00000000 00000000      .....      STRN0=02
0070  00000000 00000000 00000000 00000000      .....
0080  00000000 00000000 00000000 00000000      .....      READ
0090  00000000 00000000 00000000 00000000      .....      ADD
00A0  00000000 00000000 00000000 00000000      .....      UPDATE
00B0  00000000 00000000 00000000 00000000      .....      BROWSE
00C0  00000000 00000000 00000000 00000000      .....      DELETE
-----
1 Help      2 Format C   3 End      4 ENDB     5 PREV     6 DataType DL
7 Page bwd  8 Page fwd  9 Caps Off 10 Top     11 Bottom  12

```

The previous screen displays the contents of the first record in both hexadecimal and character formats. It is clear from looking at the character display that the data string THIS IS NOT A NAME has not been moved to the beginning of this record.



Note: Setting the No File Updating option prevented the demo program from updating this file, *without* affecting the demo program's source code.

Before returning to the demo program, we will discuss some of FILE's capabilities.

FILE Facility

The FILE facility lets you view, update, add and delete records, and search for character strings. You can perform these functions at any time (for example, while your program is at a breakpoint). You can use FILE with VSAM and BDAM files, DL/I and DB2 databases, temporary storage records, and transient data records. You can also use FILE when no program is executing to perform routine file maintenance.

FILE displays records in dump format, character format, vertical format, and structured format. To display the record in different formats, press PF2. For structured format, which displays records or DL/I segments on a field-by-field basis, you also must identify the program containing the structure.

Example

Take advantage of the FILE facility when you are testing your own programs. You can use FILE to change your test records or to create additional records in the middle of a test session. You can also use FILE to make sure your program has successfully updated a file.

The Demo Program Completes Execution

1. Press Clear.
CA InterTest for CICS redisplay the File Selection screen.
2. Type =X in the Command field and press Enter to return to the Source Listing Breakpoint screen.
3. Press PF5 to resume program execution.
CA InterTest for CICS returns you to the demo program's Options Menu.

Review What Happened

In this part of the demo session you took advantage of several CA InterTest for CICS features. You were able to:

- Use a request breakpoint to halt the demo program prior to the first File Control REWRITE command.
Request breakpoints make it easy for you to halt your program at various points, such as before all HANDLE CONDITION commands or all Terminal Control commands. When your program halts, you can inspect main storage or auxiliary storage to detect errors and review program logic.
- Use the No File Updating option to prevent a program from updating a file.
Preventing a program from updating a file means you can test a program repeatedly without having to recreate test data. Many programs can share the same test file because no program will actually change it. A test program can even use a production file without corrupting it.
- Display a record in a file.
Displaying a record in a file lets you see whether your program is working correctly. The FILE facility also lets you add, delete, and update records to meet your individual testing needs and maintain files without special one-time programs.

This concludes Option 03 of the advanced demo session. Press Enter to continue with the demo session or press Clear to terminate the session.

Option 04 How to Detect a Storage Violation

This section of the test session details how CA InterTest for CICS detects and prevents a storage violation and lets you continue testing.



Important! To perform this section, you should have completed the steps outlined earlier in this article in [Demo Preliminaries \(see page \)](#) . The CA InterTest for CICS Demo Session Options Menu should be displayed.

1. Select option 04, and press Enter.

The demo program displays a screen that describes what will occur in this part of the test session.

```

*****
****                                     ****
****                               CA InterTest Demo Session          ****
****                         Storage Violation Detection Option      ****
****                                     ****
*****
    
```

The program attempts to move data from one field to another. However, because the program has relinquished storage for the receiving field, a storage violation would occur. Instead, here is what will happen:

1. CA InterTest halts the program at an automatic breakpoint.
2. You inspect the FREEMAIN instruction which released the storage for the field.
3. You return to the breakpoint display. Then you go around the problem by continuing program execution from the statement following the one that triggered the breakpoint.

When this section of the demo completes, you will return to the Options Menu.

Press ENTER to continue or CLEAR to terminate

2. Press Enter.

CA InterTest for CICS halts the demo program at an automatic breakpoint.

Prevent a Storage Violation

The next screen is a breakpoint display screen. CA InterTest for CICS halted the demo program at an automatic breakpoint. CA InterTest for CICS prevented the storage violation that would have resulted if the highlighted MOVE statement had been executed. CA InterTest for CICS has also inserted an A next to the statement to indicate an automatic breakpoint.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= PL1DEMO  Option #      Stmt #                      Margin= 01
                        Search=
-----
----- CKPTNAME                | PROTCPF
----- NUP_AREA                 |          THIS IS NOT A NAME PRIME THE
----- REC_LEN                   |          +00681.
----- INPUT_REC_2              |          .....
----- OUTPUT_REC_2            |          .....
-----+-----
- 1431.1    1  1                END;
- 1432.1    1                    OUTPUT_REC_2    = INPUT_REC_2;
A ==>.1     1                    INPUT_REC_2    = OUTPUT_REC_2;    ==>
==> an attempt to change an area that does not belong to this task.
==> Possible system damage has been prevented.
==>
==>          Press PF1 for a detailed description.
==>

- 1434.1    1                    GOTO SENDMP00;
- 1435.1    1                    LOOPRTN:
- 1436.1    1                    DO I = 1 TO 50;
- 1437.1    1                    /* EXEC CICS ASKTIME */
    
```

Preventing storage violations is one of the most important reasons to monitor programs. Storage violations are notorious CICS errors that can:

- Bring down your CICS system.
- Cause a program to produce erroneous data.
- Cause one program to corrupt the data of another program. In this case, tracking down the problem without CA InterTest for CICS is almost impossible because the error is not in the affected program.

By stopping the program before the storage violation can occur, CA InterTest for CICS protects your system and helps you diagnose and correct the error.

In this example, the statement that would have caused the storage violation is:

```
INPUT_REC_2 = OUTPUT_REC_2
```

This statement moves data from OUTPUT_REC_2 TO INPUT_REC_2. The program may not own the area of storage defined as INPUT_REC_2, so the program cannot move data to that field.

Inspect the FREEMAIN Statement

1. Page back to find out what caused the error.
2. Press PF7.

The following screen appears.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>>
Program= PL1DEMO Option # Stmt # Search= Margin= 01
-----
----- CKPTNAME | PROTCPF
----- NUP_AREA | THIS IS NOT A NAME PRIME THE
----- REC_LEN | +00681.
----- INPUT_REC_2 | .....
----- OUTPUT_REC_2 | .....
-----
+-----
- 1416.1 1 STORVIOL:
- 1417.1 /* EXEC CICS GETMAIN SET(INPUT_REC_PTR)
- 1418.1 LENGTH(6144) */
- 1419.1 DO;
- 1420.1 1 1 DCL DFHENTRY_BF6577CC_B36530EB BASED(ADDR(DFHEI0)
- 1421.1 ASSEMBLER) ENTRY(*,POINTER, FIXED BIN(15));
- 1422.1 1 1 CALL DFHENTRY_BF6577CC_B36530EB(' . . . . 0462000
- 1423.1 00 03 00 00 8C 00 F0 F4 F6 F2 F0 F0 F0 F0 'X */ ,INPUT_R
- 1424.1 1 1 END;
- 1425.1 /* EXEC CICS FREEMAIN DATA(INPUT_REC) */
- 1426.1 1 DO;
- 1427.1 1 1 DCL DFHENTRY_BF6577CC_B36C7BE6 BASED(ADDR(DFHEI0)
- 1428.1 ASSEMBLER) ENTRY(*, *);

```

Look at the FREEMAIN statement that releases the storage for INPUT_REC.

Now it is clear why the storage violation occurred. The demo program released the storage for INPUT_REC. After releasing the storage, the program tried to move data to INPUT_REC_2, which is part of INPUT_REC.

To correct the problem, you have to remove the FREEMAIN statement or change its location, and then recompile the program. However, one of the benefits of CA InterTest for CICS is that you do not have to correct a problem when you discover it. Instead, you can go around the problem and continue testing the program.

Now we will redisplay the breakpoint.

3. Press Clear.
CA InterTest for CICS redisplay the automatic breakpoint.

Resume Program Execution

Continue program execution at the first instruction after the statement that triggered the breakpoint. In the PL/I demo, this is GOTO SENDMPO0.

1. Tab to the GOTO SENDMPO0 statement following the breakpoint.
2. Type G (go).
3. Press Enter to continue program execution.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>>
Program= PL1DEMO Option # Stmt # Search= Margin= 01
-----

```

```

----- CKPTNAME                    | PROTCPF
----- NUP_AREA                    |          THIS IS NOT A NAME  PRIME THE
----- REC_LEN                      | +00681.
----- INPUT_REC_2                  | .....
----- OUTPUT_REC_2                 | .....
-----+-----
- 1431.1    1  1                    END;
- 1432.1    1                      OUTPUT_REC_2    = INPUT_REC_2;
A ==>.1     1                      INPUT_REC_2    = OUTPUT_REC_2;    ==>
==> an attempt to change an area that does not belong to this task.
==> Possible system damage has been prevented.
==>
==>          Press PF1 for a detailed description.
==>

g 1434.1    1                      GOTO SENDMP00;
U 1435.1    1                      LOOPRPTN:
- 1436.1
- 1437.1                      DO I = 1 TO 50;
                              /* EXEC CICS ASKTIME */

```

CA InterTest for CICS resumes program execution. PL1DEMO successfully completes execution and the Options Menu appears. By going around the storage violation, you have allowed PL1DEMO to continue executing despite the problem CA InterTest for CICS detected.

Review What Happened

In this part of the demo session:

- CA InterTest for CICS detected and prevented a storage violation and halted the demo program at an automatic breakpoint.
- CA InterTest for CICS' ability to prevent storage violations protects your CICS system and prevents one program from corrupting another.
- By inspecting the statement that would have caused the storage violation, you saw which field was in error. By inspecting the FREEMAIN statement, you were able to determine why the program did not own the storage area.
- Using CA InterTest for CICS to examine the source code surrounding an error makes it easy to learn what caused the problem.
- You went around the problem and continued program execution.
- The ability to go around a problem lets you continue testing and correct many errors in a single test session.

This concludes Option 04 of the advanced demo session. Press Enter to continue with the demo session or press Clear to terminate the session.

Option 05 How to Test a Composite Module

This option details how you can use CA InterTest for CICS to test and debug a composite module. Composite support is a CA InterTest for CICS feature designed to help programmers who are responsible for testing and debugging called subroutines. Skip this option if you are not responsible for these subroutines.

To perform this option, you should have set monitoring for the composite module as described earlier in this article in [Composite Support \(see page 245\)](#).

In this section you will test three programs:

- The main program, PL1DEML, which like PL1DEMO, is invoked by the DEMP transaction.
- Two subroutines called by PL1DEML:
 - ASBIN25, written in Assembler
 - PSBIN25, written in PL/I

PL1DEML has several subroutines; however, you need to test only two of them.

Similarly, composite modules at your site might have many called subroutines, but you need to tell CA InterTest for CICS only about the ones you want to test.

1. Display the CA InterTest for CICS Demo Session Options Menu.
2. Select option 05, and press Enter.

The demo program displays a description of what will occur in this part of the demo.

```

*****
****                                     ****
****                               CA InterTest Demo Session          ****
****                               Composite Support                    ****
****                                     ****
*****
    
```

Program PL1DEML calls two subroutines: ASBIN25 and PSBIN25.

1. CA InterTest halts the program at an automatic breakpoint caused by an ASRA. Without composite support, you will see only that you are stopped at the CALL to ASBIN25.
2. You abend the task, and then set composite support for PL1DEML.
3. Type in transaction id DEMP again. This time you will see the actual instruction causing the ASRA within the subroutine.
4. You correct the problem that caused the automatic breakpoint.
5. You then set an unconditional breakpoint in PSBIN25. When you resume execution, the program is halted at that breakpoint.
6. You remove the breakpoint and the program completes execution.

When this section of the demo completes, you will return to the Options Menu.

Press ENTER to continue or CLEAR to terminate

3. Press Enter.

The composite module halts at an automatic breakpoint at statement CALL ASBIN25.



Note: If a screen other than the one below appears, someone at your site has already set composite support for the composite module.

```

CA InterTest for CICS - PROTSYM FILE  ABEND DETECTED BREAKPOINT
COMMAND ===>
    
```

```

Program= PL1DEML  Option #      Stmt #      Search=      Margin= 01
-----+-----
_ 1644.1      1              GOTO END_SESSION;
_ 1645.1
A ==>.1      1      CALLCOMP:      ==>
==> CICS abend intercepted. Abend code = ASRA. EXEC CICS request was
==> LINK. Abend in program ASBIN25 at offset 000000E8.
==>
==>      Press PF1 for a detailed description.
==>

_ 1647.1              /* EXEC CICS LINK PROGRAM ('PL1DEML') */
_ 1648.1              DO;
_ 1649.1      1  1      DCL DFHENTRY_BF6577CC_B52431E6 BASED(ADDR(DFH
_ 1650.1      NTER ASSEMBLER) ENTRY(*,CHAR(8)));
_ 1651.1      1  1      CALL DFHENTRY_BF6577CC_B52431E6('000011100000
_ 1652.1      00000000000110000000000000000000000010000000011110000111
_ 1653.1      11011011110000111100001111000011110000'B /* '0E 02 80 00
_ 1654.1      F0 F5 F9 F6 F0 F0 F0 F0 'X */, 'PL1DEML');
_ 1655.1      1  1      END;
_ 1656.1      1              GOTO SENDMP00;

```

CA InterTest for CICS halted the composite module at an automatic breakpoint because it detected and prevented an error in ASBIN25 when control passed to that subroutine. However, so far you do not know *where* in ASBIN25 the error occurred. Notice that the name in the Program field is PL1DEML, the name of the composite module. This is the program CA InterTest for CICS is monitoring; the program for which you set monitoring before you began the advanced demo session. Setting composite support lets you test and debug a called subroutine as if it were a separate program, with *full symbolic support*. This means you can set breakpoints and other monitoring options for that subroutine, and when CA InterTest for CICS detects an error, it generates an automatic breakpoint at the statement that triggered the abend, *not* at the CALL. You can set composite support before you begin to test a program or at any time during testing. In this case, you will abend the task, set composite support, and then resume testing.

Abend the Task

When your program is stopped at a breakpoint, you can abend the task rather than resume execution.

1. Type ABEND in the Command line of the Breakpoint display, and press Enter.

 **Note:** This is the same as selecting option 3 Abend, from the Breakpoint Menu. Because you do not need a dump, you do not need to type an abend code.

The following screen appears.

```

-----CA InterTest for CICS  ABEND BREAKPOINTED TASK  -----
COMMAND ==>

Type an abend code, then press ENTER.

Abend Code  ____  Abend code options are:

                blanks      Normal abend, no dump
                XXXX        Abend exits cancelled, no dump

```

```

                                your code    Your abend code, dump taken
.
.
.

```

2. Press Enter.
CICS informs you the task has been abended.
3. Press Clear.

Set Composite Support

To set composite support, access the main CNTL menu from which you can set all of CA InterTest for CICS' monitoring options.

1. Type ITST 2.1 on a clear screen, and press Enter.
This is the fastpath entry from CICS to invoke the Main Menu and select the Program Monitoring Option.
The Program Monitoring Menu appears.

```

-----CA InterTest for CICS PROGRAM MONITORING -----
COMMAND ==>

    Type information and S to set or R to remove option(s) below.

    Program . . PL1DEML_      Program name (or .ALL, .OPTIONS or generic)
    User ID  . . _____   User (or .ANY) for whom the program is monitored

    Option      Description                                          More:  +
    - Status    Display and/or remove monitoring options (S only)
    - Monitor   Monitoring (R removes monitoring and all options previously set)
    - UBP       Unconditional breakpoints (specific program only)
    - CBP       Conditional breakpoints (specific program only)
    - RBP       Breakpoints for CICS, DB2, DL/I or external CALL requests
    - Stmt Trace Statement tracing and data monitoring (COBOL only)
    - New copy  Fetch new copy of program and reset monitoring options (S only)
    - Commands  Indirect commands defined for a specific COBOL or PL/I program
    - Replace   CICS resource name replacement options
    - Protect   Storage protection monitoring options
    - Special   Other options (storage allocation, file updating, etc.)
    s Composite Monitor multi-CSECT program's separately compiled components

    PF1 Help    2          3 End      4 Return    5          6
    PF7 Backward 8 Forward  9          10         11         12

```

2. In the Program field, type the name of the composite module you are using, PL1DEML.
3. Type S to the left of Composite, and press Enter.
The Composite Support menu appears.

```

          CA InterTest for CICS - Composite Support Builder
COMMAND ==>
Define composite support for PL1DEML and press PF5.          SCROLL: PAGE
CAIN2620 Not all subprograms are selected                    Row 00001 of 00006

* *      *      *      *      *      *
_ Link name Monitor  Offset  Length  Language  Comments
S PL1DEML1 PL1DEML   80     248    PL/I     4.3
S PSBIN251 PSBIN25   EE8    1B0    PL/I     4.3
S ASBIN25  ASBIN25  1098   17C    HLASM    6.0
_ @PL1DEML  _____ 1218   10
_ PLIXOPT  _____ 1228   8
_ @PSBIN25  _____ 1230   8

```

PF1 Help	2	3 End	4	5 Process	6 RFind
PF7 Backward	8 Forward	9	10	11	12

This menu displays link-edit information for the composite program and its subroutines. A batch job, executed after the composite module was link-edited, gave CA InterTest for CICS this information.

Three *monitor-names* are listed: PL1DEML, ASBIN25, and PSBIN25. These are the names under which CA InterTest for CICS monitors the main program and two of the subroutines.

The batch job that provides the link-edit information identifies all of a program's called subroutines. For efficiency, however, it is a good idea to specify monitor-names just for the subroutines you want to test. You can always set monitoring for additional subroutines by assigning monitor-names on the Composite Support menu.

4. Press PF5 to set composite support.

CA InterTest for CICS sets composite support for PL1DEML and returns you to the Program Monitoring menu.

Now CA InterTest for CICS monitors the subroutines and the composite module PL1DEML so you can take advantage of all CA InterTest for CICS facilities to test and debug *each* of the programs.

Re-execute the Demo Program

Now you can re-execute the demo program.

1. Press PF4.
The Primary Option Menu appears.
2. Select option X Exit to exit to CICS.
3. Clear the screen and type the transaction ID DEMP, and press Enter.
4. Press PF2 for the Options Menu.
Within the Options Menu, choose Option 05 (Composite Support) and press Enter.
5. Press Enter again at the explanation screen.
CA InterTest for CICS displays the automatic breakpoint where you initially halted. However, now the breakpoint occurs at the *actual statement* in ASBIN25 that triggered the abend, not at the CALL to ASBIN25.



Note: The name in the Program field is ASBIN25. When control passes to ASBIN25, CA InterTest for CICS begins monitoring that program because you set composite support.

```
CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= ASBIN25  Option #      Stmt #      Displacement=      Margin= 01
```

```

Search=
-----+-----
  Loc  Object Code  Addr1 Addr2  Stmt      Source Statement
- 0000E2
A 0000E2 FA20 D1A0 2143 001A0 00143 ==>      AP   TASKNUM,=P'1'
  ==>
  ==> ASRA ABEND (0C7) detected and prevented. Caused by invalid decimal
  ==> arithmetic data format.
  ==>
  ==>      Press PF1 for a detailed description.
  ==>
- 0000E8 F920 D1A0 2144 001A0 00144 367      CP   TASKNUM,=P'2'
- 0000EE 4720 20FC      000FC 368      BH   RETCICS
- 0000F2 F920 D1A0 2143 001A0 00143 369      CP   TASKNUM,=P'1'
- 0000F8 4780 214E      0014E 370      BE   RETURN
- 0000FC      372 RETCICS DS   0H
  373 *      EXEC CICS RETURN
- 0000FC 4110 D068      00068 374      DFHECALL =X'0E080000080000
  384 *
- 000118      385      LTORG
- 000118 00000000      386      =V(DFHEAI0)
  
```

Setting composite support enables CA InterTest for CICS to identify and display the statement in the subroutine where the error occurred.

The error occurred because of improperly formatted data in TASKNUM. The AP instruction tries to move a packed decimal 1 to TASKNUM. TASKNUM is defined as a packed decimal field, but it contains binary zeros instead of a valid packed decimal value. This is the same type of error you corrected in the basic demo session.

Display the Data in TASKNUM

To correct this error, display the contents of TASKNUM and dynamically change the value.

- Tab to the AP instruction, and overtype the A with D. Position the cursor under any character in TASKNUM, and then press Enter.
CA InterTest for CICS displays the contents of TASKNUM.
On the left of the screen are the Assembler data field names; on the right are the corresponding data-type appropriate representations of each field, note that fields that contain incorrect data for their type have a question mark in the first digit (TASKNUM).
CA InterTest for CICS displays more than just the contents of TASKNUM. It displays TASKNUM and all the fields below it in the same data structure; that is, in the same DSECT.

Correct the Data in TASKNUM

As you can see, TASKNUM contains binary zeros rather than a packed decimal value. You can correct the error by overtyping the incorrect value with a correct value.

- Tab to the top line and move the cursor to the question mark in the value portion for TASKNUM.
- Overtyping the current value with all zeros.
- Press Enter.

```

CA InterTest for CICS - MAIN STORAGE UTILITY - Termid = U030

Starting at Address = 101DA8  Disp  Structure Display Format
TASKNUM                1A0  | ?00000. |
TSTEXT                 1A3  | .....  |
                       | .....  |
                       | .....  |
                       | .....  |
  
```

```

TSQLEN          1EC +0006553600.
MSGNAME         1F0 0000000000.
MSGLLEN         1F4 00000.
RECRBA          1F6 .....
RECPONT         1FC 0000000000.
-----
PF1 Help        2          3 End          4 Return      5          6 Dump
PF7 Backward    8 Forward  9 Caps Off  10          11 Redisplay 12 Structure
CORE='R13.TASKNUM'
CAIN4713 R13 not found - have assumed its register value

```

CA InterTest for CICS redisplay the screen, changing your entries to uppercase. Now TASKNUM contains a packed decimal zero. You have corrected the error that triggered the automatic breakpoint.

- Press Clear.
CA InterTest for CICS redisplay the previous Source Listing Breakpoint screen. However, the explanation of the abend no longer appears.

Set a Breakpoint in Subroutine PSBIN25

Now you are going to set a breakpoint at the first statement in subroutine PSBIN25, the other subroutine you asked CA InterTest for CICS to monitor. Then, when you resume execution, CA InterTest for CICS halts execution at the breakpoint in this subroutine.

When you are stopped in a subroutine, you can use all of these CA InterTest for CICS features:

- View the program listing and compiler output
- Display and modify main and auxiliary storage
- Set and remove breakpoints and other monitoring options
- Request single-step execution
- Display the backtrace that brought the program to its current point
- Tab back to the Program field.
- Overtyping ASBIN25 with PSBIN25 and press Enter.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==>
Program= psbin25 Option # Stmt # Displacement= Margin= 01
Search=
-----+-----
Loc Object Code Addr1 Addr2 Stmt Source Statement
- A 0000E2 FA20 D1A0 2143 001A0 00143 ==> CONTINUE DS 0H
_ 0000E8 F920 D1A0 2144 001A0 00144 367 AP TASKNUM,=P'1'
_ 0000EE 4720 20FC 000FC 368 BH RETCICS
_ 0000F2 F920 D1A0 2143 001A0 00143 369 CP TASKNUM,=P'1'
_ 0000F8 4780 214E 0014E 370 BE RETURN.
.

```

CA InterTest for CICS displays the source listing for PSBIN25.
Now you will set an unconditional breakpoint at the first statement in PSBIN25.

- Tab to the first statement.
- Type **U** to the left of that statement, and press Enter.

```

CA InterTest for CICS - PROTSYM FILE ABEND DETECTED DISPLAY
COMMAND ==>
Program= PSBIN25  Option #      Stmt #                      Margin= 01
                        Search=
-----
u   18.1              PSBIN25: PROC(INPUT_REC_PTR, STOR_SIZE) OPTIONS(REENTRA
-   19.1              DCL 1 DFHCNSTS STATIC, /* CONSTANTS USED BY TRANSLATOR
-   20.1              2 DFHLDVER CHAR(22) INIT('LD TABLE DFHEITAB 530.
-   21.1              2 DFHEIB0 FIXED BIN(15) INIT(0),
-   22.1              2 DFHEID0 FIXED DEC(7) INIT(0),
-   23.1              2 DFHEICB CHAR(8) INIT(' ');
-   24.1              1 DCL DFHEPI ENTRY, DFHEIPTR PTR;
-   25.1              1 DCL 1 DFHEIBLK BASED (DFHEIPTR),
-   26.1              02 EIBTIME FIXED DEC(7),
-   27.1              02 EIBDATE FIXED DEC(7),
-   28.1              02 EIBTRNID CHAR(4),
-   29.1              02 EIBTASKN FIXED DEC(7),
-   30.1              02 EIBTRMID CHAR(4),
-   31.1              02 EIBFIL01 FIXED BIN(15),
-   32.1              02 EIBCPOSN FIXED BIN(15),
-   33.1              02 EIBCALEN FIXED BIN(15),
-   34.1              02 EIBAID CHAR(1),
-   35.1              02 EIBFN CHAR(2),
-   36.1              02 EIBRCODE CHAR(6),

```

CA InterTest for CICS sets an unconditional "before" breakpoint at the statement.

- Press Clear.
CA InterTest for CICS redisplay the previous Source Listing Breakpoint screen.

Resume Program Execution

Now you are going to continue program execution.

Press PF5.

The program resumes execution. When the control passes to subroutine PSBIN25, CA InterTest for CICS halts the program at the unconditional breakpoint you just set.

At this point you could use CA InterTest for CICS to examine the program listing of PSBIN25 or the values of any program variables. However, will remove the breakpoint and resume program execution.

Remove the Breakpoint and Resuming Execution

Overtyp the U with X, and press PF5.

CA InterTest for CICS removes the unconditional breakpoint and program execution resumes. PL1DEML completes execution and the Options Menu is displayed.

Review What Happened

In this part of the demo session:

- When you executed PL1DEML, CA InterTest for CICS detected and prevented an abend caused by invalid data in one of its called subroutines.
CA InterTest for CICS displayed the automatic breakpoint at the statement which called the subroutine, rather than the statement that triggered the breakpoint, because CA InterTest for CICS was not monitoring the subroutine.
- You set composite support for PL1DEML so that CA InterTest for CICS would monitor its subroutines: ASBIN25 and PSBIN25.
Setting composite support lets you test and debug a called subroutine with full symbolic support and take advantage of all CA InterTest for CICS features.
- When you re-executed PL1DEML, CA InterTest for CICS redisplayed the automatic breakpoint, this time at the actual statement in subroutine ASBIN25 that caused the error.
- You corrected the error by overtyping the contents of the invalid field.
- You then set an unconditional breakpoint in the other subroutine, PSBIN25, and resumed program execution. When control passed to PSBIN25, CA InterTest for CICS halted execution at the breakpoint you set.
The ability to halt execution at any statement in a subroutine means you can test a subroutine just as if it were a main program.
- You removed the unconditional breakpoint and continued program execution.

This concludes Option 05 of the advanced demo session. Press Enter to continue with the demo session or press Clear to terminate the session.

CA InterTest Batch Demo Sessions

The CA InterTest Batch Demo sessions are designed to train new users in the basic product features used to test and debug programs. These demos also introduces some of the advanced features of CA InterTest for CICS.

We recommend that all CA InterTest Batch users take these demo sessions and become familiar with the features explained here. For complete information about these and other features, see [CICS Debugging \(https://docops.ca.com/display/CAITSD11/CICS+Debugging\)](https://docops.ca.com/display/CAITSD11/CICS+Debugging) .

Getting Help

The Help facility is online documentation of product features. It makes it easy to learn and use the product. Help is available from all product screens.

To view an online summary of new features for this release, use ITST Option 8 What's New.

Basic Foreground Demo Session

This section 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.

- [Demo Session Objectives \(see page 273\)](#)
- [Before You Start the Demo \(see page 274\)](#)
- [Begin the Basic Demo \(see page 274\)](#)
- [Access the Primary Option Menu \(see page 275\)](#)
- [File Allocation \(see page 276\)](#)
- [Specify the Program to be Tested \(see page 276\)](#)
- [Begin Program Execution \(see page 278\)](#)
- [Detect and Intercept an Abend \(see page 278\)](#)
- [Determine the Cause of the Error \(see page 279\)](#)
- [Dynamically Change the Value in Tasknum \(see page 280\)](#)
- [Control Program Execution \(see page 281\)](#)
- [Set Unconditional Breakpoints \(see page 282\)](#)
- [Remove the Breakpoint and Keep Window \(see page 284\)](#)
- [Continue Execution \(see page 285\)](#)
- [What You Have Learned \(see page 285\)](#)

Demo Session Objectives

This demo session, teaches you 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 article 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 article cannot anticipate differences at your installation, you need to know certain information specific to your product installation before attempting the examples in this demo.

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.CAVHSAMP. 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 [Advanced Demo Session \(see page 289\)](#). 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 article illustrate a COBOL II demo session using CAMRCOBB. The screen images of other demos vary from the panels shown in this article, but the steps for performing the demo are the same.

When testing a program such as CAMRCOBB 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 SOC7.

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.] IBALLOD  
[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:

- **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.CAVHJCL(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 about allocating DDs to your TSO session or converting JCL to ALIB format, see [Allocations Facility for ISPF](https://docops.ca.com/display/CAITSD11/Allocations+Facility+for+ISPF) . (<https://docops.ca.com/display/CAITSD11/Allocations+Facility+for+ISPF>)



Note: For instructions on converting JCL to CA Roscoe RPF format, see [JCL Conversion for CA Roscoe](https://docops.ca.com/display/CAITSD11/JCL+Conversion+for+CA+Roscoe) (<https://docops.ca.com/display/CAITSD11/JCL+Conversion+for+CA+Roscoe>).

Specify the Program to be Tested

Begin by selecting the Foreground option.

Follow these steps:

1. In the Primary Option Menu, type 1 and press Enter.
The Execution Control Panel displays. This is where you specify the name of the application to be tested.
2. 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.
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.
3. Enter the name of the demo program and the PROTSYM file that was used when the program was compiled.
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 article reference the COBOL II program CAMRCOB2.

The statement line numbers shown here 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 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-TIME0 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 *

```

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, type the FREQ command and 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-TIME0 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.

Follow these steps:

1. Type the GO command and Press Enter.
CAMRCOB2 begins execution and displays the "Welcome Panel" panel.
2. Press Enter to continue the demo.
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.

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 ==>
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,

```

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 \(see page 289\)](#). For now, assume that zero was the value intended, correct the value of Tasknum, and continue our testing.

Type the following command: set tasknum = 0, and press Enter.

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.

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 or just after the statement is executed. (Depending on the type of unconditional breakpoint that you set.)
- 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 [Advanced Demo Session \(see page 289\)](#).

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 U line command, the) line command, or the UNCOND control command. The U command sets a breakpoint that stops *before* the statement is executed. The) command sets a breakpoint that stops *after* the statement is executed. The UNCOND command lets you specify whether the breakpoint stops before or after the statement is executed.

See how to set a breakpoint before the execution of the DISPLAY-2ND-PANEL using the U line command.

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.

```

The application resumes program execution and continues until it reaches the breakpoint that you just set.

```

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.

```

The application halts the program before the first executable statement in the paragraph is executed and displays the screen.

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 "before" 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. 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.

Type the following command to remove the Keep Window: remove all. Tab down and type over the U with an X to remove the breakpoint and 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.

```

The application removes the Keep Window and the unconditional breakpoint.



Notes:

- 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

Follow these steps:

1. Type GO to continue program execution.
CAMRCOB2 next displays a screen confirming that you have successfully corrected the problem in the demo program.
2. Press Enter.
CAMRCOB2 displays the following demo summary screen.
3. To complete this part of the sample test session, press Enter.
CAMRCOB2 displays the demo exit screen.
4. Exit the demo program by pressing Enter.
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.

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.

Basic Batch Link Demo

This section 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.

- [Demo Session Objectives](#) (see page 286)
- [Before You Begin](#) (see page 286)
- [Begin the Batch Link Demo](#) (see page 286)
- [Test Your Programs Using Batch Link](#) (see page 288)
- [What You Have Learned](#) (see page 289)

The basic demo session illustrates many of the testing and debugging tasks you use on your own programs. For more information about these tasks, see [Test Your Programs Using Batch Link](#) (see page 288).

Demo Session Objectives

This demo session builds on what you have learned in [Basic Foreground Demo Session](#) (see page 273). The focus in this section 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.CAVHSAMP. 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.CAVHJCL(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 [Batch Link Facility \(https://docops.ca.com/display/CAITSD11/Batch+Link+Facility\)](https://docops.ca.com/display/CAITSD11/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.CAVHSAMP
//INT1LOAD DD DISP=SHR,DSN=CAI.CAVHLOAD
//INT1PNLL DD DISP=SHR,DSN=CAI.CAVHPNL1
//INT1MSG1 DD DISP=SHR,DSN=CAI.CAVHMSG0
//INT1PROF DD DISP=SHR,DSN=CAI.PROFLIB
//INT1CLIB DD DISP=SHR,DSN=CAI.CAVHSRC
//INT1CLOG DD SYSOUT=*
//INT1REPT DD SYSOUT=*
//STEPLIB DD DISP=SHR,DSN=CAI.CAVHLOAD
// 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 Link Selection -----
COMMAND INPUT ==> * * * * * SCROLL ==> PAGE
* * * * * * * * * * *
# Jobname Stepname Program Owner User Trans SYSID
***** TOP OF DATA *****
1 JOB01A STEP1 TESTPGM USER01 TS001
2 JOB02A STEP1 CAMRCOBB USER02 TS002
3 JOB03X RUN PAYROLL USER03 LINKED BY USER03 TS001
***** 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 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 SOC7 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 about the Utility option, see [CA InterTest Batch Utilities \(https://docops.ca.com/display/CAITSD11/CA+InterTest+Batch+Utilities\)](https://docops.ca.com/display/CAITSD11/CA+InterTest+Batch+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 [Batch Link Facility \(https://docops.ca.com/display/CAITSD11/Batch+Link+Facility\)](https://docops.ca.com/display/CAITSD11/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.

Advanced Demo Session

While [Basic Foreground Demo Session \(see page 273\)](#) and [Basic Batch Link Demo \(see page 285\)](#) 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:

- **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.



Note: Before accessing the Demo Session Options Menu, you must complete the setup steps outlined in the [Advanced Demo Preliminaries \(see page 290\)](#) . You also need to know if the names of the following libraries have been changed at your installation:

- CAVHPROC
- CAVHSAMP
- ALIB
- INCLIB

Advanced Demo Preliminaries

The advanced demo session uses the same program used in the basic demo session: CAMRASM for Assembler users, CAMRCOB2 or CAMRDMDR2 for COBOL II and higher users, CAMRCOB or CAMRDMDR 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 \(see page 290\)](#)
- [Split Your Screen \(see page 291\)](#)
- [Advanced Demo Data Sets \(see page 291\)](#)
- [INT1CLIB \(see page 293\)](#)

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, perform the following actions:

- 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))
```

1. Submit the job to create the data set.
2. Check the condition code of the job 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 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
ALLOC INT1REPT DSN=userid.INT1REPT DISP=SHR
```

- **CA Roscoe users**

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.CAVHSAMP, 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.

Follow these steps:

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.CAVHSAMP') SHR
```

- **CA Roscoe users**

```
ALLOC INT1CLIB DSN=CAI.CAVHSAMP DISP=SHR
```

- 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 \(see page 294\)](#)
- [Access the Demo Session Options Menu \(see page 295\)](#)
- [Exit the Advanced Options Demo \(see page 295\)](#)
- [Return to the Advanced Options Demo Session \(see page 296\)](#)
- [Option 1 Time-Controlled Execution \(see page 296\)](#)
- [Option 2 Set Conditional Breakpoints \(see page 298\)](#)
- [Option 3 Update Source Code \(see page 300\)](#)
- [Option 5 Trace Program Execution \(see page 307\)](#)
- [Option 6 Work with Indexed Table Items \(see page 310\)](#)
- [Option 7 Histogram Report \(see page 313\)](#)

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 \(see page 84\)](#) you set an unconditional breakpoint by using the U or) line command, which are both 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.CAVHSAMP contains the application commands that set the breakpoints required for using the Advanced Options Demo for COBOL. Include this data set member as follows:

Type the following command at the *Initial* Intercept panel: **include demoincl** and press Enter. 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 ==>>
TRACE=> 000876
          000875      PROCEDURE DIVISION USING PARAMETER-AREA.
*- -> 000876      START-PROGRAM.
*- -> 000877      OPEN OUTPUT REPORT-OUT.
          000878      * -----*
          000879      * Welcome to the <productname> 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.

Follow these steps:

1. From the Initial Intercept panel, type GO and press Enter to execute the program. The demo program begins execution and displays the Welcome panel.
2. Press PF2 to access the Options Menu for the Advanced Demo Session. The Advanced Options Preliminary Screen displays, reminding you not to access the Options Menu without including the data set member DEMOINCL.
3. Press Enter to continue. 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.

Follow these steps:

1. Press PF3. The End Demo Session screen displays.
2. Press Enter. The Program Exit Intercept panel displays.
3. Type GO and press Enter to execute the program. 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 steps in [Access the Demo Session Options Menu \(see page 295\)](#). 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 \(see page 290\)](#) 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.

Type Option 1 and Press Enter.

CAMRCOB2 displays the panel that describes what occurs in this part of the demo session.

Press Enter.

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.

Type the following command: k ytd; k x-axis.

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: 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.

```
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.

Type the following command: slow 2 and press Enter.

The application initiates a time-controlled execution of CAMRCOB2, at a rate of one verb every two seconds.

When the value of X-AXIS in the keep window is MAR, press the ATTN or PA1 key. Try the ATTN key first.

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.

Type the following command: remove all and press Enter.

The application removes the keep window for YTD and X-AXIS on the Attention Intercept panel.

Continue Execution

Type GO and press Enter to continue execution.

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 article](#) , 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 \(see page 290\)](#) and you have the Demo Session Options Menu displayed on your screen.

Follow these steps:

1. Type Option 2 and press Enter.
CAMRCOB2 displays the screen that describes what occurs in this part of the demo session.
2. Press Enter.
The application halts CAMRCOB2 at an unconditional breakpoint. The first executable statement in the paragraph is highlighted.
3. 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.
4. Type the COND command and press Enter.
The application displays the When panel. WHEN is a synonym for COND.

5. Make the entries on the When panel and Press Enter.

The application displays the following message in the top right corner of the When panel:
WHEN CONDITION SET.

```
----- 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
```

6. Press PF3 to exit the When panel.

The application brings you to the previous Breakpoint Intercept panel.

7. Continue execution by typing GO and pressing Enter.

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.

```
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 D$PLY,
001113 COBDPN73.
--> 001114 MOVE 4 TO COBDPN71-LENGTHS.
--> 001115 CALL 'ISPLINK' USING V$COPY,
001116 V$COPY-COBDPN71,
```

A sample Intercept panel showing condition YTDCOND has been met follows:

Now remove the conditional breakpoint before continuing.

Follow these steps:

1. Using the WHEN command to return to the When panel.
The When panel displays.
2. Type the following values on the When panel and press Enter:
OPTION: ==> r
WHEN NAME ==> ytdcond
The application displays the following message in the top right corner of the When panel:
WHEN DELETED.
3. Press PF3 to exit the When panel.
The application brings you to the previous Intercept panel.

Remove the Keep Window and Continue Execution

Remove the keep window before continuing execution.

Follow these steps:

1. Type the REMOVE ALL command and Press Enter.
The keep window is removed.
2. Type GO and press Enter.
CAMRCOB2 resumes execution and displays the Options Menu.

This concludes Option 2: Conditional Breakpoints. Select another option and go to the appropriate section, 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 \(see page 290\)](#) and you have the Demo Session Options Menu displayed on your screen.

Follow these steps:

1. Type Option 3 and press Enter.
CAMRCOB2 displays the screen that describes what occurs in this part of the demo session.
2. Press Enter.
The application halts CAMRCOB2 at an unconditional breakpoint at the paragraph SPLIT-SCREEN.

This concludes Option 3: Update Source Code. Select another option and go to the appropriate section, 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 \(see page 290\)](#) 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.

Follow these steps:

1. Key in Option 4. Press Enter.
CAMRCOB2 displays the screen that describes what occurs in this part of the demo session.
2. Press Enter.
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.

3. Continue execution by typing GO and pressing Enter.
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.
4. Halt the program to see what is going on. 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.
The Attention Intercept panel displays.

Continue with the [Continuing from the Attention Routine \(see page 304\)](#) 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, follow the instructions in the next session to continue.

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 [Continuing from the Attention Routine \(see page 304\)](#) .

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 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.

Follow these steps:

1. Type the following command to set the keep window: k sub-6 and press Enter.
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.

```

2. Type the following command to execute CAMRCOB2 one verb every two seconds: slow 2 and press Enter.

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.

3. Halt execution again by pressing the ATTENTION or PA1 key. After the Attention Routine message displays, press Enter. The application brings you to another Attention Intercept panel. The current statement is highlighted.
4. 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.
5. Tab down to the statement: MOVE 1 TO SUB-6, and type 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 *

```

The application skips the execution of the MOVE statement, and executes the code as if the MOVE statement was removed.

6. Tab up to the first statement in the TOTAL-ORDER routine, ADD 1 to SUB-6, and type 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 *

```

7. Press Enter.

CAMRCOB2 resumes execution from the line where you typed 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.

Follow these steps:

1. Type the REMOVE ALL command.
2. Tab down and overtype the S to the left of the MOVE statement with an X and press Enter. The keep window is removed, and the skip line command is removed.
3. Continue execution by typing GO and pressing Enter. 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, 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 \(see page 290\)](#) and you have the Demo Session Options Menu displayed on your screen.

Follow these steps:

1. Type Option 5 and press Enter.
CAMRCOB2 displays the screen that describes what occurs in this part of the demo session.
2. Press Enter.
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.

Type PREV 9 in the Command field and press Enter.
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.)

Type the A command and press Enter.

The application highlights the next statement executed.

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.

```

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.

Follow these steps:

1. Type the CS command and Press Enter.

The application displays the original Breakpoint Intercept panel shown in the beginning of this option.

2. Type the GO command and press Enter.
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, 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 \(see page 290\)](#) and you have the Demo Session Options Menu displayed on your screen.

Follow these steps:

1. Select Option 6 and press Enter.
CAMRCOB2 displays a screen that describes what occurs in this part of the demo session.
2. Press Enter.
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),
0018 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 SOC7 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 typing K to the left of the highlighted statement.

Tab to the highlighted ADD instruction, type K (for keep) and press Enter. 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.

Type the following 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),

```

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.

Type the STEP1 command and press Enter.

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),

```

Execute CAMRCOB2 by entering the command: GO and pressing Enter.

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.

Type the following command: step 0; remove all

Press Enter.

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.

Follow these steps:

1. Type the STEP command and press Enter.
The application displays the value of the step count in the top-right corner; in this case, the message is: NO STEP COUNT SET.
2. Resume execution of CAMRCOB2 by entering GO.
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, 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 \(see page 290\)](#) and you have the Demo Session Options Menu displayed on your screen.

Follow these steps:

1. Type Option 7 and press Enter.
CAMRCOB2 displays the screen that describes what occurs in this part of the demo session.
2. Press Enter.
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.

Type the HIST command and press Enter.

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.

Follow these steps:

1. ISPF users

Tab a few lines down your screen and press the SPLIT PF key defined in your ISPF profile (normally PF2).
The ISPF Main Menu displays in the split-screen area.

CA Roscoe users

Enter the CA Roscoe SPLIT sequence.
A second CA Roscoe panel displays.

2. 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 \(see page 290\)](#) . 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.
The first page of the histogram report displays. A sample report is shown in the following panel.



Note: In the following example, the data set name is: intbatch.histrept.

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.

Type the M command and press PF8.

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.

ISPF users

Remove your split-screen using the command: X

CA Roscoe users

Use the CA Roscoe END key to end the second session.
 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.

Type the FS1311 command and press Enter.
 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.

Follow these steps:

1. Type the CS command and press Enter.
 The application displays the Program Exit Intercept panel.
2. To continue the advanced demo, enter the command go options and press Enter. To end the demo, enter the command GO.
 If you are continuing the demo, CAMRCOB2 program execution continues with the display of

the Advanced Options Menu. Choose another option from the menu. 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.



Note: Before proceeding, be sure you have completed the steps outlined in [Advanced Demo Preliminaries \(see page 290\)](#) and you have the Demo Session Options Menu displayed on your screen.

Advanced Demo Session for Assembler

This article illustrates an Assembler advanced demo session using CAMRASM. Before proceeding, be sure you have completed the steps outlined in [Advanced Demo Preliminaries \(see page 290\)](#) and you have the Initial Intercept for CAMRASM displayed on your screen.



Note: CAMRASM is an ISPF application.

- [Include Unconditional Breakpoints \(see page 317\)](#)
- [Access the Demo Session Options Menu \(see page 318\)](#)
- [Exit the Advanced Options Demo \(see page 318\)](#)
- [Return to the Advanced Options Demo Session \(see page 319\)](#)
- [Option 1 Time-Controlled Execution \(see page 319\)](#)
- [Option 2 Set Conditional Breakpoints \(see page 321\)](#)
- [Option 3 Update Source Code \(see page 323\)](#)
- [Option 4 Interrupt a Looping Program \(see page 324\)](#)
- [Option 5 Trace Program Execution \(see page 328\)](#)
- [Option 7 Histogram Report \(see page 329\)](#)

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 or) line command, which are both 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.CAVHSAMP contains the application commands that set the breakpoints required for using the Advanced Options Demo for Assembler. Include this data set member as follows:

Type the following command at the *Initial* Intercept panel:

```
include asmincl
```

Press Enter.

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 ==>
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'
*-->          00000C F0F561F1F661F0F3          20          DC          CL9'&SYSDATE'
                                +          DC          CL9'05/16/03'
*-->          000015 F1F74BF5F340          21          DC          CL6'&SYSTIME'
                                +          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.

Follow these steps:

1. From the Initial Intercept panel, type GO and press Enter to execute the program.
The demo program begins execution and displays the Welcome panel.
2. Press PF2 to access the Options Menu for the Advanced Demo Session.
The Advanced Options Preliminary Screen is displayed, reminding you not to access the Options Menu without including the data set member ASMINCL.
3. Press Enter to continue.
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-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.

Follow these steps:

1. Press PF3.
The End Demo Session screen displays.
2. Press Enter.
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.)
3. Type GO and press Enter to execute the program.
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 in [Access the Demo Session Options Menu \(see page 318\)](#). 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 \(see page 290\)](#), 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.

Follow these steps:

1. Type Option 1 and press Enter.
CAMRASM displays the screen that describes what occurs in this part of the demo session.
2. Press Enter.
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.

Type the following command: k ytd; k x_axis_ytd

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                                442          STH    R0,SUB_4
--> 000420                                444 PAY_CALC_LOOP DS 0H
--> 000420 45E0 C440                                445          BAL    R14,POPULATE_GRAPH
--> 000424 4810 D102                                447          LH     R1,SUB_4
--> 000428 4110 1001                                448          LA     R1,1(,R1)
--> 00042C 4010 D102                                449          STH    R1,SUB_4
--> 000430 4910 C9CE                                451          CH     R1,NO_OF_HOURS
--> 000434 4740 C420                                452          BL     PAY_CALC_LOOP
--> 000438 45E0 C7EC                                454          BAL    R14,UPDATE_CHEQUE
--> 00043C 47F0 C316                                456          B     OPTIONS
--> 000440                                458 POPULATE_GRAPH DS 0H
--> 000440 4820 D102                                459          LH     R2,SUB_4
--> 000444 4C20 CDF4                                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.

Follow these steps:

1. Type the SLOW 2 command and press Enter.
CA InterTest Batch initiates a time-controlled execution of CAMRASM, at a rate of one statement every two seconds.
2. When the value of X_AXIS_YTD in the keep window is MAR, press the ATTN or PA1 key. Try the ATTN key first.
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                                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                                467          MH     R2,=Y(L'X_AXIS_ITE
0003 00045A 4122 D65B                                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                                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:

Type the `REMOVE ALL` command and press Enter.

The application removes the keep window for YTD and X_AXIS_YTD on the Attention Intercept panel.

Continue Execution

Type `GO` and press Enter to continue execution.

CAMRASM resumes execution, and displays the Options Menu.

This concludes Option 1: Time Controlled Execution. Select another option and go to the appropriate section, 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 \(see page 290\)](#) and you have the Demo Session Options Menu displayed on your screen.

Follow these steps:

1. Type Option 2 and press Enter.
CAMRASM displays the screen that describes what occurs in this part of the demo session.
2. Press Enter.
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.

Follow these steps:

1. Type the COND command in the Command field and press Enter.
The application displays the When panel. WHEN is a synonym for COND.
2. Make the following entries on the When panel and 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
    
```

The application displays the following message in the top right corner of the When panel:
WHEN CONDITION SET.

3. Press PF3 to exit the When panel.
The application brings you to the Intercept panel.
4. Continue execution by typing GO and pressing Enter.
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.

Follow these steps:

1. Return to the When panel by entering the WHEN command and pressing Enter.
The When panel displays.
2. Type the following on the When panel and press Enter:

```
OPTION: ===> r
```

```
WHEN NAME ===> ytdcond
```

The application displays the following message in the top right corner of the When panel:
WHEN DELETED.

3. Press PF3 to exit the When panel.
The application brings you to the previous Intercept panel.

Remove the Keep Window and Continue Execution

Remove the keep window before continuing execution.

Type the REMOVE ALL command and press Enter.
The keep window is removed.

Return to the Options Menu by continuing execution.

Type GO and press Enter.
CAMRASM resumes execution and displays the Options Menu.

This concludes Option 2: Conditional Breakpoints. Select another option and go to the appropriate section, 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 \(see page 290\)](#) and you have the Demo Session Options Menu displayed on your screen.

Follow these steps:

1. Type Option 3 and press Enter.
CAMRASM displays the screen that describes what occurs in this part of the demo session.
2. Press Enter.
The application halts CAMRASM at an unconditional breakpoint at the label SPLIT_SCREEN.
3. Tab about halfway down your screen and press the SPLIT PF key defined in your ISPF profile (normally PF2).
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 SOC7 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

Follow these steps:

1. Remove your ISPF split-screen using the command: =X
You return to a full-screen display of the unconditional breakpoint shown in the previous panel.
2. Type GO and press Enter to continue execution.
You return to the Demo Options Menu.

This concludes Option 3: Update Source Code. Select another option and go to the appropriate section, 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 \(see page 290\)](#) 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.

Follow these steps:

1. Type Option 4 and press Enter.
CAMRASM displays the screen that describes what occurs in this part of the demo session.
2. Press Enter.
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.

3. Continue execution by typing GO and pressing enter.
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.
4. 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.
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 next 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 that follows.

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 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.

Follow these steps:

1. Type the following command to set the keep window: k sub_6. Press Enter.
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 *

```

2. Type the SLOW 2 command to execute one CAMRASM statement every two seconds and press Enter.
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.
3. Halt execution again by pressing the ATTENTION or PA1 key. After the Attention Routine message displays, press Enter.
The application brings you to another Attention Intercept panel. The current statement is highlighted.
4. 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.
5. Tab down to the statement: 'MVC SUB_6,ONE' and type S to the left of the line.
The application skips the statement that sets SUB_6 to 1, and executes the code as if the statement was removed.
6. Back up to the first statement in the TOTAL_ORDER procedure, 'LH R2,SUB_6', and type G to the left of the line, to continue execution from that point.
Press Enter.
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,=(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.

Follow these steps:

1. Type the REMOVE ALL command. 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 X. Press Enter. The keep window is removed, and the skip line command is removed.
2. Continue execution by typing GO and pressing Enter. 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, 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 \(see page 290\)](#) and you have the Demo Session Options Menu displayed on your screen.

Follow these steps:

1. Type Option 5 and press Enter. CAMRASM displays the screen that describes what occurs in this part of the demo session.
2. Press Enter. 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.

Follow these steps:

1. Type the PREV 9 command in the Command field and press Enter.
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.)
2. Type the A command and press Enter.
The application highlights the next statement executed.
3. Repeat the Advance step a few more times.
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.

Follow these steps:

1. Type the CS command and press Enter.
The application displays the original Breakpoint Intercept panel shown in the beginning of this option.
2. Type the GO command and press Enter.
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, 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 \(see page 290\)](#) and you have the Demo Session Options Menu displayed on your screen.

Follow these steps:

1. Type Option 7 and press Enter.
CAMRASM displays the screen that describes what occurs in this part of the demo session.

2. Press Enter.
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.
3. Type the HIST command and press Enter.
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.

Follow these steps:

1. Tab a few lines down your screen and press the SPLIT PF key defined in your ISPF profile (normally PF2).
The ISPF Main Menu displays in the split-screen area.
2. 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 \(see page 290\)](#). The suggested data set name was: userid.INT1REPT.
The first page of the histogram report displays.
3. Page down to the end of the report using the maximum command with PF8.
4. Type the M command. Press PF8.
The last page of the histogram report displays.
5. Remove your split-screen using the command: X.
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.

Type the FS 902 command and press Enter.
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.

Follow these steps:

1. Type the CS command and press Enter.
The application displays the Program Exit Intercept panel.
2. To continue the advanced demo, enter the command go options and press Enter. To end the demo, enter the command GO.
If you are continuing the demo, CAMRASM program execution continues with the display of the Advanced Options Menu. Choose another option from the menu. 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

This article illustrates a PL/I advanced demo session using CAMRPLI. Before proceeding, be sure you have completed the steps outlined in [Advanced Demo Preliminaries \(see page 290\)](#) and you have the Initial Intercept for CAMRPLI displayed on your screen.



Note: CAMRPLI is an ISPF application.

- [Include Unconditional Breakpoints \(see page 331\)](#)
- [Access the Demo Session Options Menu \(see page 332\)](#)
- [Exit the Advanced Options Demo \(see page 333\)](#)
- [Option 1 Time-Controlled Execution \(see page 333\)](#)
- [Option 2 Set Conditional Breakpoints \(see page 335\)](#)
- [Option 3 Update Source Code \(see page 338\)](#)
- [Option 4 Interrupt a Looping Program \(see page 339\)](#)
- [Option 5 Trace Program Execution \(see page 345\)](#)
- [Option 5 Trace Program Execution \(see page 348\)](#)
- [Option 6 Work with Indexed Table Items \(see page 350\)](#)
- [Option 7 Histogram Report \(see page 354\)](#)

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 or) line command, which are both 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.CAVHSAMP contains the commands that set the breakpoints required for using the Advanced Options Demo for PL/I. Include this data set member as follows:

Enter the following command at the *Initial* Intercept panel: include pliincl

Press Enter.

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.
*/
/*-----*/
SEQUENTIA      2      1      0      DCL REPORT FILE RECORD
RECSIZE(80));  3      1      0      DCL 1 LINE_OUT, ENV (F,
                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.

Follow these steps:

1. In the Initial Intercept panel, type GO and press Enter to execute the program. The demo program begins execution and displays the Welcome panel.
2. Press PF2 to access the Options Menu for the Advanced Demo Session. The Advanced Options Preliminary Screen is displayed, reminding you not to access the Options Menu without including the data set member PLIINCL.
3. Press Enter to continue. The Demo Session Options Menu is displayed.

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. 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.

Follow these steps:

1. Press PF3.
The End Demo Session screen is displayed.
2. Press Enter.
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.)
3. Type GO and press Enter to execute the program.
The demo program ends, and the Entry panel is displayed.

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 steps in [Access the Demo Session Options Menu \(see page 332\)](#). 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 \(see page 290\)](#) 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.

Follow these steps:

1. Select option 1 and Press Enter.
CAMRPLI displays the screen that describes what occurs in this part of the demo session.

2. Press Enter.

The application halts CAMRPLI at an unconditional breakpoint set at procedure PAY_CALC. The procedure statement is highlighted.

3. 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. Type the following command: k ytd; k x_axis_ytd
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 ==>
TRACE=> 000168 000094 000093 000092 000091 000088 000087 000086 000085 000084
0

-----
          1F1A84EF NP-S 000090 02 YTD                                +00000000.00
          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;

-->      174   2  0          /* CHEQUE_LINE                      = ' ';          */
-->      175   2  0          CHEQUE_AMOUNT                      = TOTALGROSS;
-->      176   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.

Follow these steps:

1. Type the SLOW 2 command and press Enter.

The application initiates a time-controlled execution of CAMRPLI, at a rate of one statement every two seconds.

2. When the value of X_AXIS_YTD in the keep window is MAR, press the ATTN or PA1 key. Try the ATTN key first.

The Attention Intercept panel displays, as shown in the following panel. A sample Attention Intercept panel follows:

```

CAMRPLI ----- CA InterTest Batch ATTENTION Intercept -----
COMMAND ==>
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.

Type the REMOVE ALL command and press Enter.

The application removes the keep window for YTD and X_AXIS_YTD on the Attention Intercept panel.

Continue Execution

Type GO and press Enter to continue execution.

CAMRPLI resumes execution, and displays the Options Menu.

This concludes Option 1: Time Controlled Execution. Select another option and go to the appropriate section, 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 \(see page 290\)](#) and you have the Demo Session Options Menu displayed on your screen.

Follow these steps:

1. Select option 2 and press Enter.
CAMRPLI displays the screen that describes what occurs in this part of the demo session.
2. Press Enter.
The application halts CAMRPLI at an unconditional breakpoint. The statement at the breakpoint is highlighted.
3. 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.
4. Type the COND command in the Command field and press Enter.
The When panel displays. WHEN is a synonym for COND.
5. Make the following entries on the When panel and 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

```

The application displays the following message in the top right corner of the When panel:
WHEN CONDITION SET.

6. Press PF3 to exit the When panel.
The application brings you to the previous Breakpoint Intercept panel.
7. Continue execution by typing GO and pressing Enter.
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.

Follow these steps:

1. Return to the When panel by entering the WHEN command and pressing Enter.
The When panel is displayed.
2. Type the following on the When panel and 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:
==>
```

The application displays the following message in the top right corner of the When panel:
WHEN DELETED.

3. Press PF3 to exit the When panel.
The application brings you to the previous Intercept panel.

Remove the Keep Window and Continue Execution

Remove the keep window before continuing execution.

Type the REMOVE ALL command and press Enter.
The keep window is removed.

Return to the Options Menu by continuing execution.

Type GO and press Enter.

CAMRPLI resumes execution and displays the Options Menu.

This concludes Option 2: Conditional Breakpoints. Select another option and go to the appropriate section, 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 \(see page 290\)](#) and you have the Demo Session Options Menu displayed on your screen.

Follow these steps:

1. Select option 3 and press Enter.
CAMRPLI displays the screen that describes what occurs in this part of the demo session.
2. Press Enter.
The application halts CAMRPLI at an unconditional breakpoint at the label SPLIT_SCREEN.
3. Tab about halfway down your screen and press the SPLIT PF key defined in your ISPF profile (normally PF2).
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

OPTION ==>                                     CCCCCCCCCCCCCCCCCCCCCCCC USER01
                                                CCCCCCCCCCCCCCCCCCCCCCCC CATS0

06/05/02

C CA Products                                CCCC                                14:09
D SVC Dump Index                            CCCC                                XAD1
                                                CCCC                                AAAAAA
    
```

```

G Color Graphics Terminal Products      CCCC      AAAAAAAAAA
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

Follow these steps:

1. Remove your ISPF split-screen using the command: =X
You return to a full-screen display of the unconditional breakpoint shown in the previous panel.
2. Type GO and press Enter to continue execution.
You return to the Demo Options Menu.

This concludes Option 3: Update Source Code. Select another option and go to the appropriate section, 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 \(see page 290\)](#) 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.

Follow these steps:

1. Select option 4 and press Enter.
CAMRPLI displays the screen that describes what occurs in this part of the demo session.
2. Press Enter.
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.

3. Continue execution by typing GO and pressing Enter. 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.
4. 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. The Attention Intercept panel displays.

Continue with the instructions in [Continuing from the Attention Routine \(see page 342\)](#) .



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 follow.

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 instructions in the [Continuing from the Attention Routine \(see page 342\)](#) section that follows.

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 ==>
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.

Follow these steps:

1. Type the K I command to set the keep window and press Enter.

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 ==>
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;
    
```

2. To execute one CAMRPLI statement every two seconds, type the SLOW 2 command and press Enter.

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.

3. Stop the execution by pressing the ATTENTION or PA1 key. After the Attention Routine message is displayed, press Enter. The application brings you to another Attention Intercept panel. The current statement is highlighted.
4. 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.
5. Tab down to the statement: I = 1, and type 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;
    
```

The application skips the statement that sets I to 1, and executes the code as if the statement was removed.

6. Tab up to the second statement in the DO loop, I = I + 1, and type 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;
    
```

7. Press Enter.

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.

Follow these steps:

1. Type the REMOVE ALL command.
2. Tab down to the I = 1 statement and type over the S to the left of the statement with X and press Enter.
The keep window is removed, and the skip line command is removed.
3. Continue execution by typing GO and pressing Enter.
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, 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 \(see page 290\)](#) and you have the Demo Session Options Menu displayed on your screen.

Follow these steps:

1. Select option 5 and press Enter.
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.

2. Press Enter.

The application halts CAMRPLI at an unconditional breakpoint that you included at the beginning 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;

0001      174   2   0          /* CHEQUE_LINE                       = ' ';          */
0001      175   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.

Follow these steps:

1. Type the PREV 9 command in the Command field and press Enter.

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.)

2. Type the A command and press Enter. The application highlights the next statement executed.
3. 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

```

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 \(see page 290\)](#) and you have the Demo Session Options Menu displayed on your screen.

Follow these steps:

1. Select option 5 and press Enter.
CAMRPLI displays the screen that describes what occurs in this part of the demo session.
2. Press Enter.
The application halts CAMRPLI at an unconditional breakpoint that you included at the beginning 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;

0001      174   2  0      /* CHEQUE_LINE                       = ' ';          */
0001      175   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 PREVIOUS command. The PREVIOUS command lets you view previously executed statements, but does not cause any statements to execute. After you scroll back using the PREVIOUS command, you can scroll forward and retrace execution using the ADVANCE command. You may specify a number from 1 to 999 with the PREVIOUS and ADVANCE commands to scroll more than one statement at a time.

The PREVIOUS and ADVANCE commands do not change the statement where CAMRPLI resumes execution; the PREVIOUS and ADVANCE only scroll the statements in the trace.

Follow these steps:

1. Type the PREV 9 command in the Command field and press Enter.

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 ( DSPY,
                                          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.)

2. Type the A command and press Enter.

The application highlights the next statement executed.

3. 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 ( DSPY,
                                          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

```

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.

Follow these steps:

1. Type the CS command and press Enter.
The application displays the original Breakpoint Intercept panel shown in the beginning of this option.
2. Type the GO command and press Enter.
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, 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 \(see page 290\)](#) and you have the Demo Session Options Menu displayed on your screen.

Follow these steps:

1. Select Option 6 and press Enter.
CAMRPLI displays a screen that describes what occurs in this part of the demo session.
2. Press Enter.
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 SOC7 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.

Press Enter to remove the ABEND message box.
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 typing K to the left of the highlighted statement.

Tab to the highlighted statement and type K (for keep).

Press Enter.

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;

---->    129    1    0    /* TASK_STRUCTURE.TASK_TEXT = ' '; */
                                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.

Type the following 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;

--->      129   1   0      /* TASK_STRUCTURE.TASK_TEXT      = ' ';          */
                                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 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.

Follow these steps:

1. Type the STEP 1 command and press Enter.

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;

--->      129   1   0          /* TASK_STRUCTURE.TASK_TEXT  = '  ';          */
                                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:

```

2. Execute CAMRPLI by entering the command: GO and pressing Enter.
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.

Follow these steps:

1. Type the following command and press Enter: STEP 0; REMOVE ALL.
The application resets the step count and removes the keep window.
2. To display the current status of the step count, enter the step command without any count number. Type the STEP command and press Enter.
The application displays the value of the step count in the top-right corner; in this case, the message is: NO STEP COUNT SET.
3. Resume execution of CAMRPLI by entering GO.
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, 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 \(see page 290\)](#) and you have the Demo Session Options Menu displayed on your screen.

Follow these steps:

1. Select option 7 and press Enter.
CAMRPLI displays the screen that describes what occurs in this part of the demo session.
2. Press Enter.
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:

          59   1  0   RELEASE ISPLINK;
          60   1  0   CLOSE FILE ( REPORT );
          61   1  0   RETCODE = DM_RC;
          62   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.

3. Type the HIST command and press Enter.
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.

Follow these steps:

1. Tab a few lines down your screen and press the SPLIT PF key defined in your ISPF profile (normally PF2).
The ISPF Main Menu is displayed in the split-screen area.
2. 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 \(see page 290\)](#). The suggested data set name was: userid.INT1REPT



Note: In the following example, the report is viewed in SDSF.

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 ---- |
    
```

3. Page down to the end of the report using the maximum command with PF8
4. Type the M command and press PF8.
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
 RYAR002 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 RYAR002
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.

5. Remove your split-screen using the command: X
 You return to a full-screen display of the Program Exit Intercept shown in the beginning of this option.
6. Use the Find Statement (FS) command to display the code that was never executed. Type the following command and press Enter: FS 240
 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);
DCL NO.  IDENTIFIER                                ATTRIBUTE AND CROS
                                                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

```

```
101          BILL_YTD                                102
                                                    /* 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.

Follow these steps:

1. Type the CS command and press Enter.
The application displays the Program Exit Intercept panel.
2. To continue the advanced demo, enter the command go options and press Enter. To end the demo, enter the GO command.
If you are continuing the demo, CAMRPLI program execution continues with the display of the Advanced Options Menu. Choose another option from the menu.
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.

Quick References

This section contains the quick reference guides for CA InterTest for CICS and CA InterTest Batch. Click on the file to view it, or use the right-click menu to save the file to your computer or print it.

[CA InterTest Batch Quick Reference Guide](#)

[CA InterTest for CICS Quick Reference Guide for Assembler, COBOL, and PL/I Users](#)

[CA InterTest for CICS Quick Reference Guide for All Users](#)