

Wednesday, May 17, 2006 9:43 am

**IBM 4764 PCI-X Cryptographic Coprocessor
ICAT Debugger
Getting Started**

Wednesday, May 17, 2006 9:43 am

Second Edition (May, 2006)

Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult with your IBM representative to be sure you have the latest edition and any Technical Newsletter.

IBM does not stock publications at the address given below. This and other publications related to the IBM 4764 PCI-X Coprocessor Card can be obtained in PDF format from the Library page at <http://www-03.ibm.com/security/cryptocards>.

Reader's comments can be communicated to IBM by sending comments and questions to the e-mail address located on the product Web site at <http://www-03.ibm.com/security/cryptocards>, or you can respond by mail to:

Department VM9A, MG81
IBM Corporation
8501 IBM Drive
Charlotte, NC 28262-8563
U.S.A.

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 2005, 2006. All rights reserved.

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Table of Contents

About This Book	1
Introducing the ICAT Debugger	1
Before You Begin	1
Minimum Hardware Requirements	1
Installation	1
Environment Variables	2
Finding Source Files	2
Limitations	2
Getting Started	3
Setting up the Coprocessor	3
Setting up the Host Computer	4
Demonstration Session	5
Starting a Debug Session	6
Using the Tool Buttons	8
Helpful Tips and Hints	8
Troubleshooting	9
Ending the Debugging Session	11
Main Window	11
Expressions Supported	12
Notices	13
Copying and Distributing Softcopy Files	13
Trademarks	13
Index	15

Wednesday, May 17, 2006 9:43 am

About This Book

This document contains information to help you install and get started with the Interactive Code Analysis Tool (ICAT) debugger supplied with the IBM 4764 Developer's Toolkit. This document is intended for use as a quick reference. For more specific details, refer to the ICAT Debugger Help available from the Help menu of the debugger.

If you need assistance from any window while using the debugger, press F1 while viewing a window or choose the Help menu.

Introducing the ICAT Debugger

ICAT is a source-level debugger that enables developers to debug applications running on an IBM 4764 PCI-X Cryptographic Coprocessor (referred to in this document as the xCrypto card). The debugger provides a graphical user interface that enables a developer to:

- Locate the current point of execution within an application and view the source that corresponds to that location.
- Examine and modify an application's state, including variables and registers.
- Set breakpoints and execute machine instructions or source statements one-by-one.
- Dump the contents of the call stack.
- Intercept and diagnose exceptions generated by an application.

The ICAT debugger (hereafter referred to in this document as the debugger) is a source-level debugger that runs on a SLES 9 (SuSE® Linux Enterprise Server 9) x86 host system.

Before You Begin

This section lists the hardware and software requirements, options that can be used when compiling and linking your program, environment variables, and the search order of source files and modules.

Minimum Hardware Requirements

- Intel® x86® 2GHz processor
- 17 MB hard disk space
- If the debugger is to communicate with the xCrypto card through a null-modem serial cable, the COM port over which the debugger communicates must work reliably at 57600 baud (that is, it should be equipped with buffered UARTs).
- If the debugger is to communicate with the xCrypto card using TCP, an Ethernet adapter is required.

Installation

ICAT is part of the IBM 4764 PCI-X Application Program Development Toolkit. For installation details, refer to the *IBM 4764 PCI-X Cryptographic Coprocessor Custom Software Developer's Toolkit Guide* and the *IBM 4764 PCI-X Cryptographic Coprocessor Installation Manual* located on the Library page of the Web site located at <http://www-03.ibm.com/security/cryptocards/>.

Environment Variables

The debugger uses environment variables to manage debugging sessions and remote communication.

Use one of the following methods to set the environment variables:

- Create and modify your own command file with the environment variables you want to set. See “Helpful Tips and Hints” on page 8 for more information.
or
- Set the environment variables in the session from which the debugger is started.

Refer to the ICAT Debugger Help (**Getting Started->Before You Begin->Environment Variables**) for a list of the available environment variables and a description of each.

Finding Source Files

The debugger searches for the source files in the following order:

1. CAT_OVERRIDE environment variable, if specified.
2. The subdirectory in which the object file generated from the source was compiled (as indicated by debug information in the executable file.)
3. Host binary path or CAT_HOST_BIN_PATH environment variable, descending subdirectories if the CAT_RECURSE_PATH environment variable is set or Recursive file searching is selected on the Remote page of the Debugger Properties window.
4. Host source path or CAT_HOST_SOURCE_PATH environment variable, descending subdirectories if the CAT_PATH_RECURSE environment variable is set or Recursive file searching is selected on the Remote page of the Debugger Properties window.
5. The current subdirectory.
6. The path defined in the INCLUDE environment variable.
7. The last specified subdirectory from the Change Text File window.
8. If the debugger cannot find the source in any of the previously mentioned locations, it prompts the user to enter the location of the required source file.

The debugger searches for executable files in the following order:

1. Current directory.
2. Host binary path or CAT_HOST_BIN_PATH environment variable, descending subdirectories if the CAT_PATH_RECURSE environment variable is set or Recursive file searching is selected on the Remote page of the Debugger Properties window.

Limitations

The debugger has the following restrictions:

- The debugger can either launch an application (that is, cause an application to be loaded into the cryptographic coprocessor and assume control of the application before any instructions in the application have been executed) or attach to an application. The earliest point at which the debugger can attach to an application (that is, assume control of the application and place it under debug) is after the application’s main entry point has been started. If you want to make certain that your application does not make progress before the debugger has a chance to attach, you must code an infinite loop at the beginning of the application (and use the debugger to change the point of execution to the statement following the loop after attaching).
- If the debugger is communicating with the xCrypto card through a serial port, no other application on the host can access that port.
- Applications to be debugged must be compiled and linked in such a way that the application executable incorporates debug information. Otherwise, the debugger cannot be used to examine and manipulate the application at the source level. However, only the copy of the application executable

that the debugger reads must have this information; the copy downloaded to the coprocessor can be reduced in size by stripping debug information from it before it is downloaded.

- To source-level debug your application, you must compile and link your application with debugging data. If you are using a version of gcc earlier than 3.0, you may use either -g or -gstabs when you compile; otherwise, you must use -gstabs. To check which version of gcc you are using, issue the following from the command line:

```
gcc -v
```

- The debugger does not handle certain coding styles well. For example, it can be difficult to debug a program that has more than one source statement on a line—the debug information available to the debugger forces the debugger to treat the entire line as a single statement. Thus, you cannot set a breakpoint on, for example, the second statement on the line nor can you step through each statement.
- If the debugger is run using the PCI communication type, and the application executes for a long time before being stopped by a breakpoint, exception, halt request, and so on, it is possible that the xCrypto card time-out will take effect and leave ICAT in an unusable state. If it is necessary to allow the application to run uninterrupted for long periods of time, the xCrypto card time-out value should be set to a value sufficiently large enough to prevent time-out.

Getting Started

This section describes how to set up the coprocessor, the host, and serial communication, how to start a debugging session, and how to end a debugging session.

Setting up the Coprocessor

To set up the coprocessor:

1. Follow the instructions for installation in the *IBM 4764 PCI-X Cryptographic Coprocessor Custom Software Developer's Toolkit Guide* to install the coprocessor and the debugger's pdaemon onto the coprocessor.
2. Follow the instructions for the xCrypto card listed below, depending on the method you want the debugger to communicate with the xCrypto card. (A communication type must be specified.)

- **PCI**

- a. Set the following environment variable to specify PCI communications.

```
export CAT_COMMUNICATION_TYPE=PCI
```

- b. Start the debugger daemon by issuing:

```
<fully qualified path>/pdaemon &
```

- **Ethernet**

- a. Issue the following commands:

```
ifconfig eth0 <IP address> netmask <appropriate netmask>
```

where appropriate netmask is usually 255.255.255.0

```
route add default gw <default gateway IP address>
```

- b. Set the following environment variable to specify communications by way of an Ethernet card.

```
export CAT_COMMUNICATION_TYPE=TCP
```

- c. Start the debugger daemon by issuing:

```
<fully qualified path>/pdaemon nnnn &
```

where nnnn is a port number, of your choosing, greater than 1024.

- **Serial**

- a. Set the following environment variable to specify serial communications.

```
export CAT_COMMUNICATION_TYPE=ASYNC_SIGBRK
```

- b. Start the debugger daemon by issuing:

```
<fully qualified path>/pdaemon &
```

3. Use the DRUID utility to load your application and start it running for debugging. The application should have an infinite loop near the beginning of the code, as recommended in the *IBM 4764 PCI-X Cryptographic Coprocessor Custom Software Developer's Toolkit Guide*.

Setting up the Host Computer

To set up the host SLES 9 Linux computer:

1. Install the IBM 4764 toolkit. The debugger is packaged as part of the toolkit in the `/xctk/<version>/bin/linux` directory (<version> is the current version number of the Toolkit - for example, 324).
2. Ensure that the `~/xctk/<version>/bin/linux` directory is in the path.
3. Set the environment variables. See "Environment Variables" on page 2 for more information.
4. Follow the instructions listed below to set up serial communication:

- **PCI**

- a. Set the following environment variables to specify PCI communications.

```
export CAT_MACHINE=N
```

where `N` is the adapter; the first adapter in the system is numbered 0.

```
export CAT_COMMUNICATION_TYPE=PCI
```

- **Ethernet**

- a. Set the following environment variable to specify communications by way of an Ethernet card.

```
export CAT_MACHINE=<IP address>:<port number>
```

where `IP address` and `port number` must match what was specified in `/user0/init.sh`

```
export CAT_COMMUNICATION_TYPE=TCP
```

- **Serial**

- a. Determine the location of the serial port device. Typical Linux installations have the first serial port as `/dev/ttyS0`, the second as `/dev/ttyS1`, and so on.

- b. Enter the following commands:

```
export CAT_MACHINE=/dev/ttyS<N>
```

where `N` is the numeric identifier for the serial port being used.

```
export CAT_COMMUNICATION_TYPE=ASYNC_SIGBRK
```

Demonstration Session

The following session demonstrates the debugger manipulating the code on the coprocessor.

1. Check the `xctk/<version>/src/samples/rte` subdirectory on your host computer. This subdirectory contains the C source files and the makefiles to make the binaries that the debugger must see on your host computer.
2. Change the `xctk/<version>/src/samples/rte/card/gcc/card.mak` makefile to ensure that the variable `DEBUG` is defined (`/DDEBUG`). This ensures that the attachment loop is included in the executable. Also, make certain that `XCTK_FS_ROOT` and `UDXTK_FS_ROOT` are set in either the makefile or the shell from which the make command is issued. Depending on the cross compiler invoked to cross compile the `rte` sample, you may need to set or change the values of `CROSS` and `GCC_NAME` to appropriately reflect the cross compiler used.
3. Make the `rteX` executable. The executable is saved at `xctk/<version>/src/samples/rte/card/gcc/rteX`.
4. Use the `xctk/<version>/src/samples/rte/host/gcc/host.mak` file to make the `hreX` executable for the host computer.
5. Copy the `rteX` executable file to the `/xctk/<version>/build/user0` directory.
6. Ensure the following conditions are met in `xctk/<version>/build/user0`:
 - a. `pdaemon`, `rteX`, and `init.sh` are located in `xctk/<version>/build/user0`.
 - b. `/xctk/<version>/build/user0/init.sh` has the following line
`export CAT_COMMUNICATION_TYPE=PCI` before `pdaemon` is launched.
 - c. `pdaemon` is launched by way of `/user0/pdaemon &`.
 - d. `rteX` is launched after `/user0/pdaemon` is launched by way of `/user0/rteX` (`init.sh` supplied with the Developer's Toolkit (`xctk`) should meet these conditions).
7. Run `make fixup`, as root, to appropriately change the permissions of certain files.
8. Use `mkfs.jffs2` to create a filesystem image to load through DRUID. From `xctk/<version>/build`, start as `mkfs.jffs2 -b -r user0 -o <filename>`.
9. Use DRUID to load the `mkfs` image onto the coprocessor, start as
`druid <fileName> <publicKey> <privateKey> [adapterNumber]`
10. Modify `seticat.sh`, on the host computer, as described in "Environment Variables" on page 2 so that the `CAT_COMMUNICATION_TYPE` and `CAT_MACHINE` environment variable reflect your environment. Now, run `seticat.sh` to set up the environment variables, run `icatpcx`, and then wait for the Launch or attach window to display. (Note for the variables to be exported properly, run as `. seticat.sh`.)
11. Enter `rteX` in the **Program** field on the **Attach** page of the Launch or attach window. Click **OK** to attach the program. (To launch, on the **Launch** page, enter the full path on the card (that is, `/user0/rteX`) and then click **OK**.) It will take some time, but the Debug Session Control window will be displayed with the `rte` application in the component list.
12. Click the plus sign located beside the path name for the `rteX` executable file in the Debug Session Control window. The path expands to display a list of functions within the file. Double-click `rte` to open the Source window for `rteX`. The function should pause within the infinite loop.
13. Select a Mixed view and notice that in the view the disassembly and the C source are mixed. Switch back to a Source view.
14. Set a breakpoint within the infinite loop and then set the debugger to Run.
15. Jump to the call to `xcAttach`. At this point, you can debug the program normally. However, if you click **Run**, the system stops on `xcGetRequest()`, because no host function has asked for service yet.
16. Run the `hreX` application from the host.
Note: You must have stepped past the call for `xcAttach` for `hreX` to run properly.
17. Display a Mixed view when the debugger hits a breakpoint. Single step the assembler code a couple of times. Switch back to a Source view. Next, set a breakpoint at the check for the return code of `xcGetRequest (if(rc < 0))`, and run again. When you hit that breakpoint, you can double-click variables, do a call stack unwind, show a Register window or a Storage window, and so on.

This concludes the demonstration session.

Starting a Debug Session

Load the program you want to debug on the coprocessor. To start the debugger:

From the Linux command prompt, enter **icatpcx** followed by one of the following parameters:

- /P+** Use program profile information (the default).
- /P-** Do not use any program profile information.

The Launch or attach window is displayed.

Note: It's usually convenient to have set any environment variables you want to specify through a command file (for example, seticat.sh) before starting the debugger. See "Environment Variables" on page 2 for more information.

To launch your program:

1. Select the **Launch** tab. The **Launch** page is displayed. This is the default page.

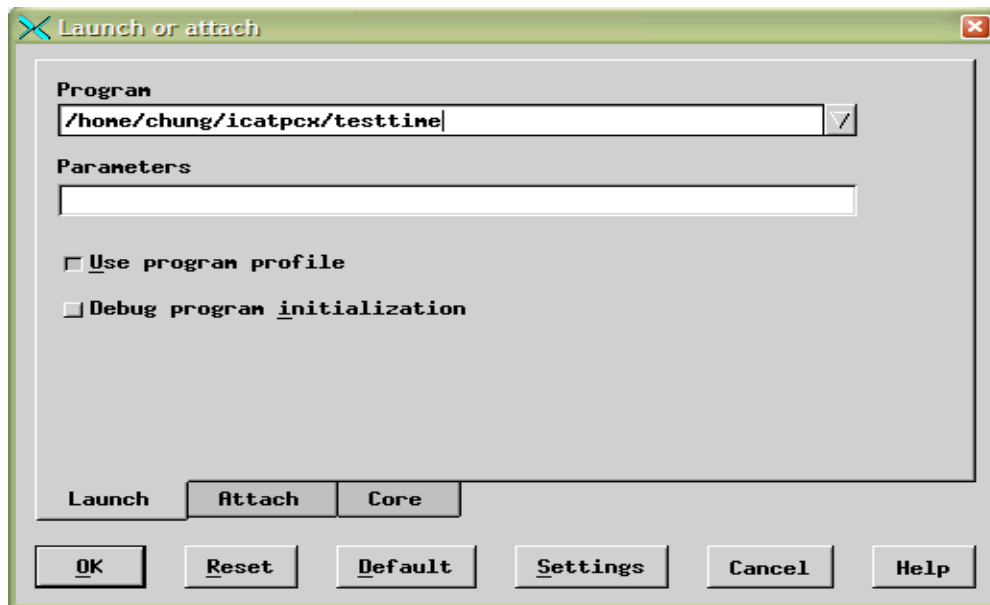


Figure 1. Launch or attach window - Launch page

2. Enter the name of the program to start in the **Program** field.
3. Enter any parameters you want to start the program in the **Parameters** field.
4. Select **Use program profile** to reactivate the windows and breakpoints, if you're going to debug a program more than once.
5. Select **Debug program initialization** if you want to debug the initialization code for the program.
6. Click **Settings** to display the **Debugger Properties** window if you want to set how threads and source files are initially displayed. Click the **Source** tab to view the **Source** page and select the changes that you want. If you make any changes, click **Apply**. Click **Close** to close the window.
7. Click **OK** to start debugging the program. The Debug Session Control window is displayed showing the threads and components of your program.

Reset returns the window settings to the values you defined upon initialization of the window.

The **Default** button restores the window's default settings.

Settings displays the Debugger Properties window, which enables you to select how threads and source files are initially displayed and enables you to set environment variables. Refer to “Setting Debugger Properties” located within the ICAT Debugger Help for more information.

To attach to a program that is running:

1. Select the **Attach** tab. The **Attach** page is displayed.

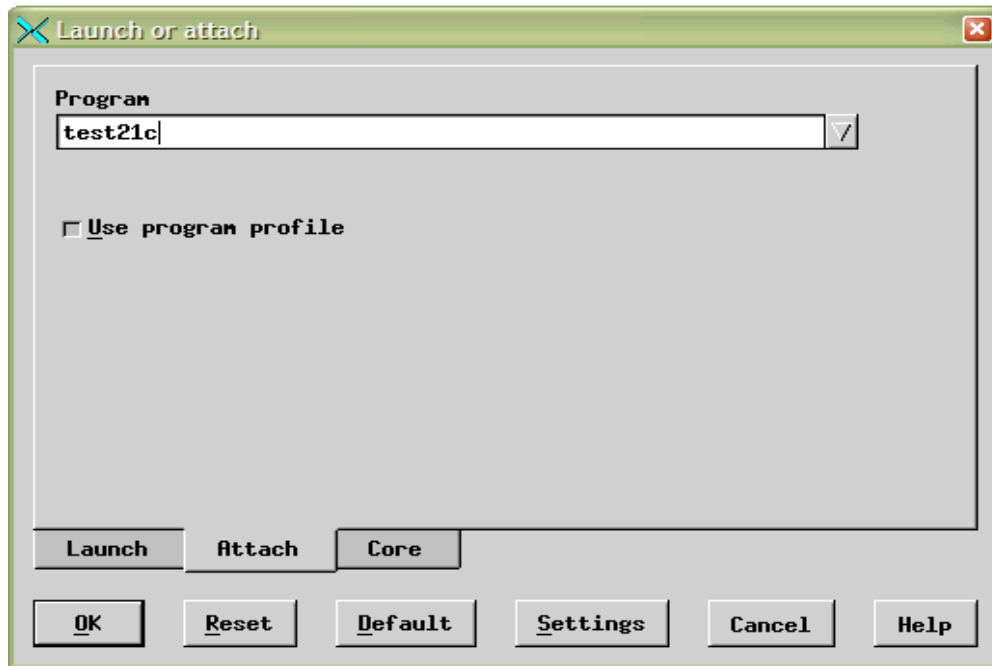


Figure 2. Launch or Attach window - Attach page

2. Enter the name of the program to start in the **Program** field.
3. Select **Use program profile** to reactivate the windows and breakpoints if you're going to debug a program more than once.
4. Click **OK** to attach to the program.

To specify the core dump file name:

1. Select the **Core** tab. The **Core** page is displayed.

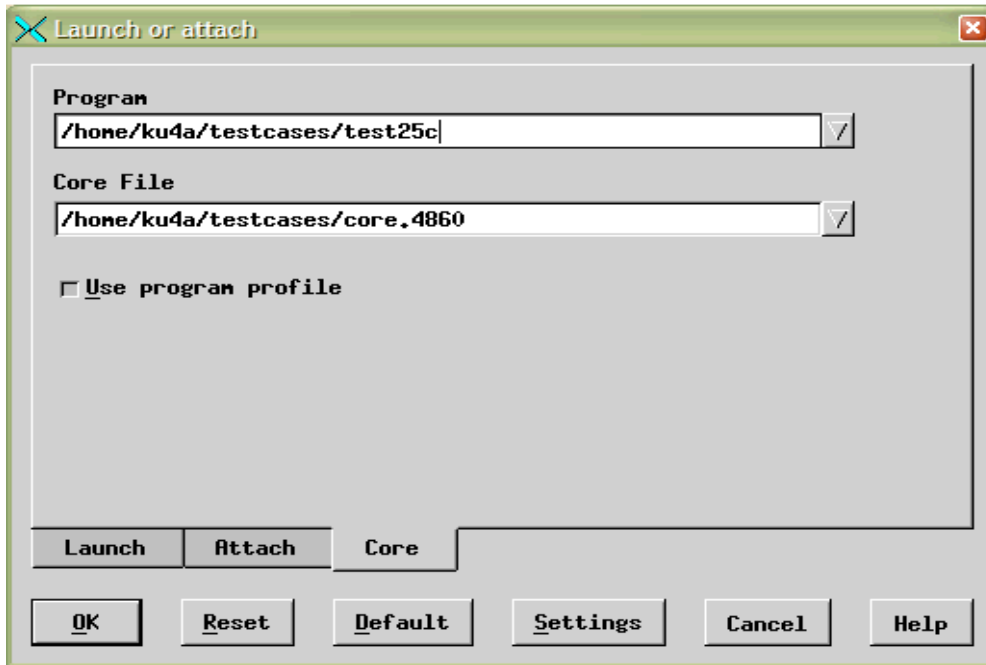


Figure 3. Launch or attach window - Core page

2. Enter the name of the program that generated the core dump file in the **Program** field.
3. Enter the name of the core dump file in the **Core File** field.
4. Select **Use program profile** to reactivate the windows, if you're going to debug a program more than once.
5. Click **OK** to start debugging the program. The Debug Session Control window is displayed showing the threads and components of your program.

Using the Tool Buttons

A tool bar has been provided on the debugger windows for easier access to frequently used features. To display buttons in a window, enable the **Tool buttons** choice that is listed under the **Options** menu. A list of the available tool buttons and their features is included in the ICAT Debugger Help (**Getting Started->Using the Tool Buttons**).

Helpful Tips and Hints

The following tips and hints may be helpful:

- You must have SLES 9 Linux on your host computer.
- Put any environment variables that you want set in a command file which you would start before running the debugger. For example:
 1. Create a file (for example, seticat.sh) which contains the environment variables you want set:

```
export ICAT_BROWSER=<path to browser>
```
 2. Ensure the command file is executable by issuing:

```
chmod +x seticat.sh
```
 3. Start the command file by using a period and a space before the filename:

```
. seticat.sh
```

4. Run the debugger by issuing:

```
icatpcx
```

- Using C, you can write your program code with stylistic features that are not supported by the debugger. For example, multiple statements on the same line are difficult to debug. None of the individual statements can be accessed separately when you set breakpoints or when you use step commands.

Troubleshooting

Use the following checklist if you're having problems starting a remote debug session.

1. Make certain that the debugger daemon and application have been loaded onto the coprocessor. See "Setting up the Coprocessor" on page 3 for details.
2. Wait a couple of minutes if you have just loaded your code onto the coprocessor. The loader may take some time to run the program to the point at which it can be attached.
3. Ensure the following if you are using the serial port to debug:
 - a. Your serial cable is a null-modem cable (a cable that connects the transmit data pin of one machine to the receive data pin of the other).
 - b. Your serial cable is securely attached to both the host computer and the target xCrypto card.
 - c. Buffered UARTs are on the host computer and that you are using 57600 baud.
4. Make certain that your communication port is enabled and powered on if you run the debugger on a notebook computer.
5. Ensure the following, if you want to launch an application, and specify the application name and arguments on the command line:
 - a. The value of the **CAT_COMMUNICATION_TYPE** environment variable is set to the communication mode you are using.

Value	Description
TCP	Ethernet communications
ASYNC_SIGBRK	Serial communications using null-modem serial cable
PCI	PCI-X bus communications

- b. The value of the **CAT_MACHINE** environment variable is set to the appropriate value for the communications mode you are using.

Communication mode	Value
Ethernet	The IP address of the adapter, followed by a colon, followed by the port number used by the daemon on the coprocessor.
Serial	The serial communication port number
PCI	The PCI adapter number

- c. You have a copy of the relevant executable files for your application on the host computer and the directory paths to the files are listed in the value of the **CAT_HOST_BIN_PATH** environment variable.
- d. You have a copy of the relevant source files for your application on the host computer and the directory paths to the files are listed in the value of the **CAT_HOST_SOURCE_PATH** environment variable.

- e. You set the **CAT_RECURSE_PATH** environment variable, if needed, to find the copy of the executable or source files for your application on the host computer.
 - f. If you need the daemon to launch the application from a directory other than the current directory for the session running the daemon, you set the **CAT_REMOTE_PATH** environment variable.
 - g. You specify the application name and arguments correctly.
6. Use the Launch or attach window, if you do not specify the application name and arguments on the command line, as follows:
- a. Click **Settings**. The Debugger Properties window is displayed.
 - 1. Click the **Remote** tab to view the Remote page.
 - 2. Ensure that the **Communication mode** field is set according the type of communications you are using.

Value	Description
TCP	Ethernet communications
ASYNC_SIGBRK	Serial communications using null-modem serial cable
PCI	PCI-X bus communications

Ethernet communications:

- Ensure that the **IP address** entry field is set to the IP address of the adapter.
- Ensure that the **Port #** entry field is set to the port number used by the daemon on the coprocessor.

Serial communications:

- Ensure that appropriate baud rate is specified in the **Baud rate** field.
- Ensure that **Communication port** field is set to the port number of the serial port with the connected null-modem serial cable.

PCI communications:

- Ensure that the **PCI** field is set to adapter number of the xCrypto card.

- 3. Make certain that you have a copy of the relevant executable files for your application on the host computer and the directory paths to the files are listed in the **Host binary path** entry field.
 - 4. Ensure that you have a copy of the relevant source files for your application on the host computer and the directory paths to the files are listed in the **Host source path** entry field.
 - 5. Select **Recursive file searching**, if needed, to find the copy of the executable or source files for your application on the host computer.
 - 6. Make certain that the **Remote binary path** is set if you need the daemon to launch the application from a directory other than the current directory for the session running the daemon.
 - 7. Click **Apply** if you made any changes on the window.
 - 8. Click **Close** to return to the **Launch or Attach** window.
- b. Ensure that you specify the correct program name in the **Program** entry field.
 - c. Ensure, if launching your application, that you specify any arguments to your application in the **Parameters** entry field located on the **Launch** page.
 - d. Make certain, if attaching to your application, that the application is currently executing on the target computer.

Ending the Debugging Session

To end the debugging session, click **Close debugger** (located within the **File** menu) from any of the debugger windows. The Close Debugger window is displayed. Select one of the following choices:

- Click **Yes** to end your debugging session.
- Click **No** to return to the current debugger window without exiting the debugger.

You can also end the debugging session by pressing F3 in any of the debugger windows.

Main Window

The Debug Session Control window is the control window of the debugger and is displayed during the entire debugging session. This window is divided into two panes: Threads and Components.

The Threads pane contains the threads, their names, and the state of the threads started by your program. To display the state of a thread, click the plus icon located to the left of the thread.

Right-click a selected item to display the Thread menu and press F1 to view help for this item.

The Components pane shows the path names of the modules that you are debugging. Right-click a selected item to display the Component menu and press F1 to view help for this item.

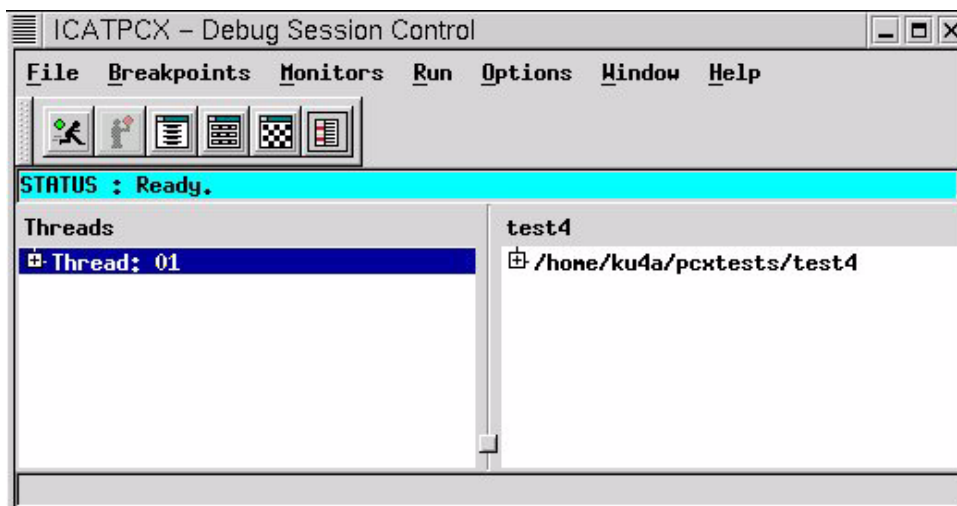


Figure 4. Debug Session Control Window

Refer to the **Windows** section of the ICAT Debugger Help for descriptions and use of the Debug Session Control window menus, and other windows available from the Debug Session Control window and a description of their menus.

Expressions Supported

The expression language supported by the debugger, which is a subset of C, includes the operands, operators, and data types.

Note: You can display and update bit fields for C code only. You cannot look at variables that have been defined using the #DEFINE preprocessor directive.

Refer to the **Expressions Supported** section of the ICAT Debugger Help for details about the expressions supported.

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights or other legally protectable rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY, 10594, USA.

Any references in this information to non- IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Copying and Distributing Softcopy Files

For online versions of this book, we authorize you to:

- Copy, modify, and print the documentation contained on the media, for use within your enterprise, provided you reproduce the copyright notice, all warning statements, and other required statements on each copy or partial copy.
- Transfer the original unaltered copy of the documentation when you transfer the related IBM product (which may be either machines you own, or programs, if the program's license terms permit a transfer). You must, at the same time, destroy all other copies of the documentation.

You are responsible for payment of any taxes, including personal property taxes, resulting from this authorization.

THERE ARE NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to you.

Your failure to comply with the terms above terminates this authorization. Upon termination, you must destroy your machine readable documentation.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both: IBM

Intel is a registered trademark of the Intel Corporation.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

SuSE is a registered trademark of SuSE Linux AG.

Other company, product, and service names may be trademarks or service marks of others.

May 17, 2006 9:43 am

Index

C

coprocessor, setting up3

D

debug session, ending11
debug session, starting6
demonstration session5

E

ending the debugging session11
environment variables2

F

finding source files2

G

getting started
 demonstration session5
 setting up the coprocessor3
 setting up the host computer4

H

hardware requirements1
helpful tips and hints8
host computer, setting up4

I

installation1
interactive code analysis tool (ICAT)1
introducing the debugger1

L

Launch or attach window6
limitations2

R

restrictions2

S

setting up the coprocessor3
setting up the host computer4
starting a debug session6

T

tool buttons8
troubleshooting9

V

variables, environment2