

Insight User Guide

Document Type	Revision	Date	Modified By	Comments
Preliminary	0.01	2004/01/13	Guevara	Initial release

1.Install.....	3
2. Using Insight.....	3
2.1 Primary interface Window.....	4
File Menu.....	4
Run Menu	4
View Menu	4
Control Menu.....	4
Preferences Menu	5
Execution Control Buttons	5
Window Buttons	5
Frame Control.....	6
2.2 How to set and view breakpoints	6
Setting a Breakpoint	6
Viewing Breakpoints	7
2.3 How to display and edit the contents of memory	7
Memory Display	7
Editing Memory.....	7
2.4 How to display and edit the contents of registers	7
Register Display	7
Editing a Register	8
2.5 View and change stack	8
Stack Display	8
Navigating the Stack Window	8
Changing Stack Levels	8
2.6 How to display and edit local variables.....	8
Locals Display	8
Editing a Variable	9
Local Variable Pop-up Menu	9
2.7 How to select target.....	9
Selecting a Target.....	9
Options	10
More Options.....	10
3.Notice.....	10
Appendix :.....	11

1.Install

- 1.Package cygwin is required to run or build Insight.It is recommended to use **cygwin** version **2.340.2.5**.
- 2.unpack the insight source package to a directory , for example /usr/src/Insight.
- 3.change into the source directory /usr/src/Insight.
- 4.You can build Insight right in the source directory . If you want build a Insight for LX5280,please configure the target as mips-lx5280-elf,if you want build a Insight for LX4181,please configure target as mips-lx4181-elf.

Example for LX5280:

```
./configure --target=mips-lx5280-elf --prefix=/usr/Insight-LX5280 --program-prefix=mips-lx5280-  ;  
make;  
make install;
```

Example for LX4181:

```
./configure --target=mips-lx4181-elf --prefix=/usr/Insight-LX5280 --program-prefix=mips-lx4181-  ;  
make;  
make install;
```

Please see the appendix for more configure options.

2. Using Insight

Just run it like you would a normal GDB (in fact, it's actually called `gdb'). If everything goes well, you should have several windows pop up. To get going,hit the Run button, and go exploring.

If you want to use Insight in command line mode, just use the **-nw** option. Or, you can undefine the DISPLAY environment variable.

Insight comes with all your standard debugger windows, including:

- o Source Window
- o Console Window
- o Register Window
- o Memory Window
- o Locals Window
- o Watch Window
- o Stack Window
- o Thread/Process Window
- o Function Browser Window
- o Debug Window (for developers)

2.1 Primary interface Window

The Source Window is the primary interface between the user and the debugger; it is automatically opened when the debugger starts. The Source Window displays the status of the program, controls execution of the program, and allows visualization of the program execution.

Its Menus look like:

File Run View Control Preferences Help

File Menu

Open

Opens a file selection dialog to select the executable to debug

Target Settings...

Opens the Target Selection Dialog to edit target settings

Page Setup

(Windows only) Opens the Windows Page Setup dialog to configure printing

Print

(Windows only) Print the contents of the Source Window Display

Exit

Exits the debugger

Run Menu

Download

Initiates download of the executable onto the target via the protocol specified in the Target Selection Dialog

Run

Runs or re-runs the program

View Menu

Stack

Open a Stack Window

Registers

Open a Register Window

Memory

Open a Memory Window

Watch Expressions

Open a Watch Window

Local Variables

Open a Locals Window

Breakpoints

Open a Breakpoint Window

Console

Open a Console Window

Function Browser

Open a window allowing the user to easily search for functions and set breakpoints.

Thread List

Open a window that displays all current threads and allows the user to change active threads

Control Menu

Step

Step program until it reaches a different source line

Next

Step program, proceeding through subroutine calls

Finish

Execute until the current stack frame returns

Continue

Continue program being debugged, after signal or breakpoint

Step Asm Inst

Step one instruction exactly

Next Asm Inst

Step one instruction, but proceed through subroutine calls

Preferences Menu

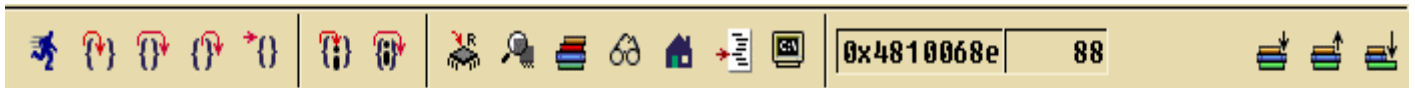
Global

Opens the Global Preferences Dialog and allows editing of global settings

Source

Opens the Source Preferences Dialog and allows editing of Source Window settings

The Source Window toolbar consists of three functional sections: execution control buttons, debugger window buttons, and stack frame control buttons.



Execution Control Buttons

These convenience buttons provide on-screen access to the most important debugger execution control functions:



or Run

The Run Button will start execution of the program, including target selection and downloading, if necessary. If the program is already running, the Run Button will start the program from the beginning (re-run it).



or Stop

The Stop Button will interrupt execution of the program (provided this feature is supported by the underlying debugging protocol and hardware) or cancel downloads. It is also used as an indication that the debugger is busy.



or Step

Step the program until it reaches a different source line



or Next

Step the program, proceeding through subroutine calls



or Finish

Execute until the current stack frame returns



or Continue

Continue the program being debugged, after signal or breakpoint



or Step Asm Inst

Step one instruction exactly. This function is only available when the Source Window is displaying assembler code.



or Next Asm Inst

Step one instruction, but proceed through subroutine calls. This function is only available when the Source Window is displaying assembler code.

Window Buttons

The Debugger Window buttons give instant access to the Debugger's auxiliary windows:















or Registers

Open a Register Window









or Memory

-  or  Stack
Open a Memory Window
-  or  Watch Expressions
Open a Stack Window
-  or  Local Variables
Open a Watch Window
-  or  Local Variables
Open a Locals Window
-  or  Breakpoints
Open a Breakpoint Window
-  or  Console
Open a Console Window

Frame Control

The Frame Control area of the toolbar displays information about the PC of the current frame, and the frame control buttons may be used to navigate through the call stack.

-  or  Up
Select and view the stack frame that called this one
-  or  Down
Select and view the stack frame called by this one
-  or  Bottom
Select and view the bottom-most stack frame

2.2 How to set and view breakpoints.

Setting a Breakpoint

Moving the mouse pointer over the "hot spot" of an executable line will change the mouse cursor to a large dot. Clicking the left mouse button will then toggle a breakpoint at this line. If no breakpoint exists, one will be installed and the dash in the left margin will change into a red breakdot. If a breakpoint exists, it will be removed and the red breakdot will revert back to a dash. The executable line marker shows the status of each line: an empty marker (the dash) indicates that no breakpoints are set at the line. A colored breakdot indicates that a breakpoint exists at the line.

Black breakdots in the Source Window display indicate that the breakpoint has been disabled. To re-enable the breakpoint, click the enable/disable checkbox in the Breakpoint Window.

Viewing Breakpoints

You can find out more information about a breakpoint by moving the cursor over a breakpoint. A balloon window will pop up with additional information. To get a list of all the active breakpoints, you



will need to open a breakpoint window.

2.3 How to display and edit the contents of memory

Memory Display

The Memory Window display is organized into a spreadsheet. The address of any cell in the Display can be determined by appending the row and column headers for the cell. Optionally, an ASCII display of the memory appears at the right. Any non-ASCII-representable byte in memory will appear in the ASCII Display as a control character (a dot, ".", by default). The Memory Preferences Dialog may be used to alter the appearance of the Memory Window.

To navigate the Memory Window, use the mouse and click the cell of interest. As an alternative, pressing the arrow keys on the keyboard will focus successive cells, from left to right, top to bottom. The focus will wrap from left to right, so hitting the right arrow key will keep advancing the address of the cell selected.

Editing Memory

To edit memory, simply enter the new value of the memory into the cell and press the enter key on the keyboard. As with the Register Window, be careful of the input format used to enter data -- the debugger is capable of parsing binary, octal, decimal, and hexadecimal values. All entries will be padded with leading zeroes, if necessary. After you hit enter, the memory window will automatically shift focus to the next cell.

To edit part of the value of a cell, you can use the mouse to position the cursor to the exact part of the value you want to change. You can also use the backspace key to delete part of the value without deleting the whole value.

2.4 How to display and edit the contents of registers

Register Display

The Register Window lists registers and their contents for the selected stack frame. It permits viewing the contents of registers in different formats, editing register values, and some display customizations.

In the Register Window, you can see all the registers on the left and their values on the right.

The Register Window will update the register contents in the display to match the stack frame currently being viewed in the Source Window and Stack Window.

Each time the program stops, the register window will automatically update. Registers that have changed since the last stop will be displayed in green.

Editing a Register

To edit a register, simply click on it with the left mouse button. Type in the new value and hit enter. You can enter a decimal, hex, or float number and the type will be converted if possible.

Press the escape key on the keyboard to cancel your edit.

2.5 View and change stack

The Stack Window allows users to view the call stack and jump between levels of the stack.

Stack Display

The Stack Display consists of a listbox which displays levels of the call stack on per line. Each line contains the level number (useful when using the Console Window) and a description of the function executing in that level. Typically, the function name and either the address of the function or the file and line number where the function is defined are displayed. The Stack Window may also be used to jump between levels of the stack.

Navigating the Stack Window

Navigation of the Stack Window is accomplished by clicking on the desired level with the left mouse button. The Source Window Display updates to show the selected frame. All other secondary windows, Registers, Watch, and Locals update their displays for the selected frame.

Changing Stack Levels

To switch frames, simply click the left mouse button on the desired frame and the debugger will switch contexts, updating all windows. The selected frame is highlighted (in gold, by default).

As an alternative, changing stack levels may be accomplished via the Frame Control Buttons on the Source Window's Toolbar. These buttons may be used to change frames one level at a time (either immediately up or immediately down) or to jump to the bottom-most stack frame.

2.6 How to display and edit local variables

The Local Variables Window displays all local variables in scope. It may be used to visualize and edit local variables. To open the Local Variables window, click on small house icon on the toolbar, or select "Local Variables" under the View pulldown menu.

Locals Display

Pointers, structures, and classes appear in the display with small expansion box before their names. To dereference pointers or view the members of classes or structures, click the closed expansion box

(which appears as a small plus sign, "+") to "expand" the listing. The expansion box changes to a minus sign, "-", indicating that the display is now open. Pointers, structures and classes may be expanded recursively to allow multiple pointer dereferences and embedded structure viewing.

The Locals Display updates after every execution of the program and highlights in green those variables whose values have changed.

The Locals Window will, by default, display all pointers in hexadecimal and all other variables in decimal. To change the display format for a variable, select the Format option from the popup-menu.

Editing a Variable

To edit a variable, either double-click the left mouse button on the value of the variable in the display or select the Edit option from the pop-up menu. To abort editing a variable's value, simply press the escape key on the keyboard. The variable's original value is restored.

Local Variable Pop-up Menu

The pop-up menu provides quick access to the functions of the Local Variables Window. To use the pop-up menu, click the right mouse button while over a variable.

- Format
Change the display format of the variable.
- Edit
Edit the variable's value.
- Delete
Remove the variable from the display.
- Dump Memory
Open a Memory Window with the variable's value as an address.
- Help
Open this help page.
- Close
Close the Local Variables Window.

2.7 How to select target

The Target Selection Dialog allows users to specify the debug target, the interface used to connect to the target, and some useful run options.

Selecting a Target

Selecting a target involves choosing a target for debugging and setting connection interface options for the target.

Common targets include: "Exec" for native debuggers, "**Remote/Serial**" for establishing a connection to a target board via a serial line, "Remote/TCP" for TCP connections, and "**Simulator**" for connections to the simulator. There may be more depending on the configuration of the debugger being used.

In general, "remote" targets are always serial connections which require the user to specify the serial port and baud rate to be used for the connection and "remote/tcp" targets are always TCP connections which require specifying the hostname and port number of the machine to which to connect. Depending upon configuration, there may be numerous serial- and TCP-based connections. These always follow the naming convention *target/Serial* and *target/TCP*.

To select a target, choose one of the available targets from the dropdown menu in the Connection

Frame. Then specify the interface options for this target: selecting the baudrate and serial port from the dropdown menus (serial targets only) or entering the hostname and port number (TCP targets only).

Options

Three run options which may be selected include:

Run until 'main'

Sets a breakpoint at main()

Set breakpoint at 'exit'

Sets a breakpoint at exit()

Display Download Dialog

Displays a dialog showing the progress of the download to the target section by section

More Options

Several additional run options may be set for each target from the Target Selection Dialog. These options govern the behavior of the debugger's Run Button. The debugger automatically selects default values for these options whenever a target is selected with the dropdown menu in the Connection Frame. To modify this default behavior, click the small triangle next to "More Options" at the bottom of the dialog. The Run Options for the current target are displayed, allowing modification of the actions for the target. When the "OK" button is selected, these settings are saved and will be used as the default for the target in future sessions.

Attach to Target

Establish a connection to the target board.

Download Program

Download the program to the target board.

Run Program

Run the program on the target board, creating a new "process". This option may not be specified along with the continue option. See note below.

Continue from Last Stop

Continue the program on the target board from where it last stopped. This option may not be specified along with the "run" option. See note below.

Note that all remote targets typically do not "run" programs. Since target boards are usually incapable of creating a new "process", these targets seldom "Run". The defaults for all remote targets reflect this distinction: they are all set to "Continue".

Only one of the options "Run Program" and "Continue from Last Stop" may be used. Typically, the default behavior of this setting should not be altered.

Insight also has an extensive online help system which describes all the windows and explains how to use them. Users are urged to browse this help system for information on using Insight . **You can select Menu “Help”->”Help Topics” to browse this help system.**

3.Notice

The stub uses instruction “break 5” to implement breakpoint . However , for LX5280,there is something special to exception handle.Different from MIPS’s precise exception,when an exception occurs,if the following two instructions write to radiax register “CBS0,CBS1,CBS2,CBE0,CBE1,CBE2”,these instructions will be finished before jumping to exception handler.If you view the registers “CBS0,CBS1,CBS2,CBE0,CBE1,CBE2”,you should pay attention to the following two instrcutions,if

there are instructions which change the “CBS0,CBS1,CBS2,CBE0,CBE1,CBE2”,what you see will be the changed values.

Please mail to zhe_jiang@realsil.com.cn if you have any questions or advice.

Appendix :

``configure' options`

Here is a summary of the ``configure'` options and arguments that are most often useful for building Insight. ``configure'` also has several other options not listed here. Note : “(configure.info)What Configure Does”,for a full explanation of ``configure'`.

```
configure  [--help]
           [--prefix=DIR]
           [--srcdir=PATH]
           [--norecursion] [--rm]
           [--enable-build-warnings]
           [--target=TARGET]
           [--host=HOST]
           [HOST]
```

You may introduce options with a single ``-'` rather than ``--'` if you prefer; but you may abbreviate option names if you use ``--'`.

``--help'`

Display a quick summary of how to invoke ``configure'`.

``-prefix=DIR'`

Configure the source to install programs and files under directory ``DIR'`.

``--srcdir=PATH'`

Warning: using this option requires GNU ``make'`, or another ``make'` that compatibly implements the ``VPATH'` feature. Use this option to make configurations in directories separate from the Insight source directories. Among other things, you can use this to build (or maintain) several configurations simultaneously, in separate directories. ``configure'` writes configuration specific files in the current directory, but arranges for them to use the source in the directory PATH. ``configure'` will create directories under the working directory in parallel to the source directories below PATH.

``--norecursion'`

Configure only the directory level where ``configure'` is executed; do not propagate configuration to subdirectories.

``--rm'`

Remove the configuration that the other arguments specify.

``--enable-build-warnings'`

When building the GDB sources, ask the compiler to warn about any code which looks even vaguely suspicious. You should only using this feature if you're compiling with GNU CC. It passes the following flags:

```
-Wimplicit
-Wreturn-type
-Wcomment
-Wtrigraphs
-Wformat
-Wparentheses
-Wpointer-arith
```

`--target=TARGET`

Configure GDB for cross-debugging programs running on the specified TARGET. Without this option, GDB is configured to debug programs that run on the same machine (HOST) as GDB itself. There is no convenient way to generate a list of all available targets.

`--host=HOST`

Configure GDB to run on the specified HOST.
There is no convenient way to generate a list of all available hosts.

`HOST ...`

Same as `--host=HOST`. If you omit this, Insight will guess; it's quite accurate.

`configure` accepts other options, for compatibility with configuring other GNU tools recursively; but these are the only options that affect Insight or its supporting libraries.