MULTI: IDE Version 8 Release Notes

Green Hills Software
30 West Sola Street
Santa Barbara, California 93101
USA
Tel: 805-965-6044
Fax: 805-965-6343
www.ghs.com

LEGAL NOTICES AND DISCLAIMERS

PubID: multi_release_notes-728459
Branch: http://toolsvc/branches/release-branch-80
Date: November 21, 2022

Contents

1. Changes in MULTI 8 1

System Requirements 2

Windows 2

Linux 2

Product Compatibility 4

Target-Specific Product Compatibility 4

New Features and Enhancements 5

Backward Trace Retrieval 5

Freeze-Mode Debug Snapshots 5

Debugging Support for C11, C++11, C++14, and C++17 Features 5

History Window Improvements 6

Trace Data Supported in the History Window 8

Python 3 Support 8

Any-Task Breakpoints Supported in TimeMachine 9

Major Performance Improvements Using the Register View Window 9

Increased Number of Supported Registers 9

Host-Side Strings Supported in MULTI Expression Evaluation 10

Target List Filtering 10

Ability to Quickly Save Different Views of a Debug Snapshot 10

Path Remapping for INDRT2 (rtserv2) Connections 10

Profile Window Improvements 11

Improved 64-Bit Linux Native Debugging 11

Support for Physical Memory Access 11

Memory Protection for Speculative Accesses 11

Pointer Expressions Column in the Memory View Window 12

Improved ROM Debugging 12

Real-Time and Arena malloc Support in Memory Allocations Window 12

Additional Call Stack Information Supported in Memory Allocations Window 13

Ability to Configure Memory Cache Usage 13
Support for Loading Multiple .mdv Files via Debug Information 13
Trace Timestamp Support on Simulators 13
Ability to Set Marks in the Editor 14
Tab Visualization in the Editor 14
Wildcard Characters Supported on Windows 14
Subversion 1.9.x Support 14
Enhanced Support for High-DPI Displays 14
Full Color Configuration Available on Windows 15
Support for Anti-Aliased Fonts in Linux 15
Local License Server (LLS) 15
Command to Display Global Symbols Using Wildcards 15
New Options to the addhook Command 15
New Options to the debugbutton Command 16
New Options to the debugsnapshot Command 16
New Option to the disassemble Command 16
New Option to the e Command 16
New Option to the l Command 17
New Option to the new Command 17
New Option to the output Command 17
New Option to the route Command 17
New Option to the sourceroot Command 17
New Options to the trace Command 18

Changes in Behavior 18
Base Class Pointers Now Cast to Derived Class Type 18
Memory Test Wizard Now Supports 64-Bit Addresses 18
Issuing the flash burn Command No Longer Prompts a Dialog 18
MULTI Waits for Run-Mode Task Initialization with the wait Command 19
Error Message When filedialog Command is Canceled 19
Ctrl+Shift+1 Now Bound to UnMark Command in Editor 19
Lines Containing ghs_noprofile Highlighted Differently in the Debugger 19
Debugger Support for Nested Functions 20
New wait Command Behavior May Require Changes to User Scripts 20
New Register View Window Configuration File 20

Register Groups Only Visible on Certain Tabs 21
Register View Tabs Displayed if Register Definition Files Are Loaded 21
Windows Restored When Loading a Debug Snapshot 21
Call Browser Replaced by the Search Window 22
print Command Outputs C and C++ Bitfields of char Type as Integers 22
Command Contexts Preserved Across Symbol Merging 22
TimeMachine Changes with SMP Targets 22
Opcode Information Provided Before Trace Decoding 23
Previously Retrieved Trace Discarded When Retrieving Trace Backward 23
Trace Processing Aborted When Disk Space is Low 23
Trace Sessions Replaced with Debug Snapshots 24
tracesave and traceload Functionality 24
Trace Options Can Be Set in the Command Pane 24
Trace Triggers No Longer Automatically Bookmarked 24
Hardware Breakpoints No Longer Synchronized Between TimeMachine and INTEGRITY Targets 25
Warning Message Removed For Shadowed MULTI Variables 25
MULTI Searches for First Matching Data Description in .mdv Files 25
Improvements to the Graphical Probe Administrator 25
DWARF Debugging Information Automatically Translated 26
Non-GUI Mode Support for the osatasklist Command 26
lic_find_licenses Utility Prints Exact License Use Time 26

Deprecated Features 26

Removed Features 27

2. Changes in MULTI 7 29

New Features and Enhancements 30
History 30
INTEGRITY TimeMachine 30
Debug Snapshot 31
Synchronous Run Control 31
Profile Window 31
Project Build Configurations 32
INTEGRITY Monolith Debugging 32

Simplified Multi-Core Configuration 32
Automatic Setup of Certain Multi-Core Targets 33
Choose Which Target List Entries are Displayed 33
Choose Which Data Explorer Entries are Displayed 33
Run to the Next/Previous Modification of a Variable 33
Set Watchpoints from the Data Explorer 34
Quickly Navigate Complex Data Structures in the Data Explorer 34
Data Explorer Column Settings Remembered Within and Across Sessions 34
Commands to Time Target Operations 34
Command to Control Breakpoint Installation After Target Setup 35
Command to List and Optionally Attach to OSA Tasks 35
Command to Clean up OSA Information or Remove OSA Support 35
Command to Modify Target-Specific Mode Variables 35
Added Options to the wait Command for Scripting 36
Added Options to the load, unload, and prepare_target Commands 36
Reduced Trace Disk Usage and Trace Session File Sizes 36
Tracing 64-bit INTEGRITY Targets Now Supported 36
Further Parallelization of Trace Processing 37
Faster Reporting of Memory Leaks and Allocations 37
Line Numbers Now Displayed in the Editor 37
Display of Task Priority, Stack Usage, and Stack Size 37
Long Target List Entries Now Easy to Tell Apart 37
Improved Target List Display of OSA Tasks on Certain Multi-Core Targets 38
New Register access Type 38
Python Interpreter Upgraded to 2.7.3 38
Subversion 1.7.x and 1.8.x Supported 38
Multiple TCP Ports Supported per License Client 38
Simpler License Manager Configuration 39

Changes in Behavior 39
Any-Task Breakpoints Set By Default in INTEGRITY 39
Watchpoints Implemented Using Any-Task Hardware Breakpoints 39

Target List Located on Left By Default 39
New Methods for Manually Dumping and Processing Profiling Data 39
Several Profiling Commands Consolidated Under the `Profile' Command 40
Continual Update of Profile Window No Longer Available 40
Data Explorer No Longer Treats Instance Variables as Local Variables 41
Data Explorer and `print /t' Now Display Variable typedef Names 41
Run-Mode Partnering Now Supported with -checksum 41
Debugger Attaches to Tasks that Hit Any-Task Hardware Breakpoint 42
Register Information Now Shared 42
Non-INTEGRITY TimeMachine No Longer Accessible from Run Mode 42
Child Windows Are Displayed in Front of the Debugger on Windows OS 43
Improved `l' Command Output 43
Issue with Windows 10 Dongle-Locked Licenses Resolved 43
Changes to Terminology Used in the Documentation 43
Stricter Error Checking on Invalid Pointer Math Expressions 43

Deprecated Features 44

Removed Features 45

3. Changes in MULTI 6.1.6 47

New Features and Enhancements 48
Support for Large Virtual Function Tables 48
Support for Freeze-Mode Debugging of INTEGRITY 48

4. Changes in MULTI 6 49

New Features and Enhancements 50
Faster Builds 50
Ability to Retrieve More Trace Data from New SuperTrace Probes 50
Faster Trace Processing 50
Separate Releases Allow Greater Flexibility When Upgrading 51

64-Bit Support in the Debugger 51
128-Bit Hardware Registers Supported 51
Security and Relationship Views Added to Integrate 51
Local Variable View Updates as You Navigate the Call Stack 51
Ability to Restore Breakpoints 52
Greater Window Availability 52
New System Variables 52
Target List Remembers Node Expansion/Collapse 52
Sortable Columns in the Target List 52
Improved Support for Preprocessor Macros in the Debugger 53
Browsing References Improvement 53
if Command Supports else if 53
Licensing Improvements 53
Faster Scanning of Subversion Checkouts 54

Changes in Behavior 54
IDE and compiler Installed into Separate Linking Directories 54
Upgrading to MULTI 6 54
License Clients No Longer Broadcast for a License Manager 54
Changes to Computer- and Dongle-Locked Licensing 54
Changes to Trace Retrieval 55
Trace Decoding No Longer Defaults to Reading Unknown Opcodes 55
Changes to TS_Packet May Require TimeMachine API Application Updates 55
Behavior of Primary and Secondary Debugger Windows 55
Large Numbers No Longer Broken Up by Default 56
Default Number of Scroll Back Lines Reduced 56
Storage of PID and TID Debugger Aliases 56

Deprecated Features 57

Removed Features 57

5. Changes in MULTI 5.0.6 59

New Features and Enhancements 60
MULTI Project Manager 60
Flexible Target Connections 60
Improved Target Management from the Debugger 60
Target Traffic Pane 61

TimeMachine Enhancements 61
Improved Kernel Object Awareness 62
Global Profiling 62
Integrated Python Engine 62
Extended Version Control Support 63
Windows Taskbar Organization 63
Improved Help Navigation 63

Changes in Behavior 63
Use of Dongle-Locked Licenses over Remote Desktop 63
Halt Issued Before Setup Script in Pre-5.0.6 Versions of MULTI 63

Unbundled Features 64

Deprecated Features 65

Removed Features 65

6. Known Issues 67

Host Support 68
IDE Components May Fail to Launch on Unsupported Linux Distributions 68
HWE Kernels Distributed with Ubuntu May Cause Problems 68
Errors from Missing DLLs in Older Windows Version 68

MULTI Installation 68
Name of Install Directory Cannot Contain Spaces 68

Project Manager Issues 69
Project Manager Performance on Very Large Project Files 69
Undefined __OS_DIR__ Macro When Opening Some ThreadX Projects 69
Projects with Build Configurations Incompatible with MULTI 6.x and Earlier 69
Truncated Field on High-DPI Displays When Creating New INTEGRITY Application Project 69

Editor Issues 70
Limitation of Brace Matching with Preprocessor Directives 70
File Permissions Under Cygwin 70

Version Control Issues 70
Subversion Authentication on Windows 70

Problems Observed with Subversion 1.4.x 71
Missing Version Information for libxml2.so.2 71
Target Connection Issues 72
Serial Ports Supported for Use with rtserv 72
Debugging Issues 72
Source Pane Displays Unexpected File 72
Maximum Number of Attached Tasks 73
Up to 65534 Macros Supported 73
Very Large Files Cause Slow Response Times When Single-/Double-Clicking Variables 73
Limitation of Browsing Traced Memory Accesses 73
Limitation of Browsing Traced Function Calls 73
Cannot Retrieve Trace During a Blocking Run or Step 74
Unexpected Debug Snapshot Behavior with trace add_text_range Command 74
Trace Configuration Files Incompatible with MULTI 4.x 74
Limitations of UNC Paths on Windows 75
Limitations for Debugging Large Enumeration Types 75
Limitation of Mixing PIC or PID Code with ABS Sections 75
Debugging AltiVec-Specific Data Types 75
Viewing Long Doubles on x86 Systems 76
`update <interval>' Not Supported with INTEGRITY TimeMachine 76
Limitations of History From RH850 Software Trace 76
Quitting During Command Line Function Call May Leave Program in Undefined State 77
Shared Memory Problems Caused by the systemd RemoveIPC Setting 77
Daylight Saving Time May Result in Erroneous Statement 77
Symbol Names for Function Templates with Template Parameter Packs 77
Compatibility with INTEGRITY 78
MULTI Ignores INTEGRITY 10 Installer Screen 78
Limitations for Loading and Unloading Applications on INTEGRITY 5.x 78
Collecting Memory Leak Information on INTEGRITY 5.x 78
Problems with Trace of Tasks in the INTEGRITY Kernel AddressSpace 79

RunTask May Not Have an Effect on Dynamically Downloaded Tasks 79
MULTI RTSERV Auto-Connect and High Priority Tasks 79
Halting or Attaching to Task Cancels a System Call with rtserv 79
System Halt with Host I/O May Make Target Unresponsive 80
Run-Mode Breakpoints May Confuse Trace 80

MULTI-Python Integration Issues 80
mpythonrun Unable to Output to stdout/stderr in Cygwin 80
MULTI-Python Integration on Linux May Require Compatibility Library 81

File System Issues 81
Tilde Expansion in Cygwin 81
Two Dots (..) Not Supported After Symbolic Links in Paths 81
NFS May Cause Poor Performance 82

Desktop Environment Issues 82
Numerous Instances of MULTI Can Exhaust the Windows Desktop Heap 82
Locking Assertion Failure May Occur Within libxcb 82
Taskbar Organizer and Windows 7 with Aero Peek 83
Taskbar Organizer and nVidia Desktop Manager 83
nVidia X Server with Compiz Window Manager May Crash 83
Drawing Problems with Exceed 2008 (13.0) 84
Nouveau Drivers for nVidia Graphics Cards May Cause Drawing Problems 84
SELinux Restricts Dynamically Loaded Code 85
MULTI May Lose X Connection After Running for Several Hours 85

Chapter 1

Changes in MULTI 8

Contents

System Requirements 2
Product Compatibility 4
New Features and Enhancements 5
Changes in Behavior 18
Deprecated Features 26
Removed Features 27

System Requirements

Note

Your Green Hills compiler may have different system requirements. For more information, see the compiler release notes.

Windows

Running a full MULTI 8 installation on Windows requires:

- A modern, multi-core x86 processor
- One of the following (64-bit only):
  - Windows 10
  - Windows 8 or 8.1
- 4 GB of RAM (16 GB is recommended.)
- 2 GB of free disk space per installation. If you are installing to a drive other than your primary Windows drive, you must also have at least 1 GB available on the primary drive for temporary files and DLLs. (100 GB of free space and a solid state drive are recommended.)
- A monitor running at a display resolution of 1024x768 or higher. (Two high-resolution monitors are recommended.)
- A DVD-ROM drive or access to the Green Hills Support site.
- To install, you must be a member of the Administrators group (or have access to the Windows System directories).

Linux

Running a full MULTI 8 installation on Linux requires:

- A modern, multi-core x86 processor
- One of the following (64-bit only):
  - Ubuntu 20.04.x LTS
  - Ubuntu 18.04.x LTS

- Ubuntu 16.04.x LTS
- Ubuntu 14.04.x LTS
- CentOS 7.x

If the IDE is linked to older probe or compiler versions, 32-bit compatibility libraries may be required. If so, follow the instructions below for your particular host.

If you are using Ubuntu 14 or later, run:

```
sudo dpkg --add-architecture i386
sudo apt-get update
sudo apt-get install libc6:i386 libncurses5:i386
libstdc++6:i386 libx11-6:i386 lib32z1
libxcursor1:i386 libxft2:i386
```

For CentOS 7.x, you may want to run the following command if the specified library is missing:

```
sudo yum install libXcursor.i686
```

If you find that the tools are not working, especially because of a licensing issue, you may need to install additional 32-bit compatibility libraries. For example, if you are using LDAP with SSS, you must install **libnss_sss**, which is usually part of the sssd-client package.

- 4 GB of RAM. (16 GB is recommended.)
- 2 GB of free disk space per installation. (100 GB of free space and a solid state drive are recommended.)
- A monitor running at a display resolution of 1024x768 or higher with true color. (Two high-resolution monitors are recommended.)
- A graphical display running the X Window System
- A mounted DVD-ROM drive or access to the Green Hills Support site
- An Ethernet device
- Write permissions to the installation directory

Product Compatibility

The MULTI 8 IDE is officially supported with:

- Green Hills 64-bit Compiler 2018.1 and later
- INTEGRITY IoT 2017.0
- INTEGRITY 11.0.2, 11.0.4, 11.2.4, 11.4.4, 11.4.6, 11.7.8, and 11.7.9
- u-velOSity 2.6.4, 2.7.1, and 2.7.2

For information about compatibility between the MULTI 8 IDE and newer versions of these products, see the release notes for the newer products.

Note

MULTI supports code built with older compilers, but the IDE will need to be linked to a compiler version 2018.1 or later.

Note

Compiler 2020.1 and older shipped with both 32-bit and 64-bit installs. MULTI 8 only supports 64-bit compilers, and it will display an error dialog if you are linked to the wrong compiler:

```
MULTI failed to load 64_cpudetect.dll:
The specified module could not be found.
```

For more information about linking the MULTI IDE to a different compiler, see "Linking IDE and Compiler/Probe Installations" in Chapter 2, "Installing Your Software" in *MULTI: Installation*.

Warning

Using MULTI 8 with a Green Hills Debug Probe requires Green Hills Debug Probe software version 6.4 or later. Do not use an earlier version of Green Hills Debug Probe software with your MULTI 8 installation.

Target-Specific Product Compatibility

- Certain targets require newer compiler versions when debugging with TimeMachine and History, which are described below.

Target	Green Hills Compiler Version
ARM64	2018.5 or later
V850	2018.5 or later (Target application can be built with any compiler version.)

New Features and Enhancements

Backward Trace Retrieval

Oftentimes, the most interesting trace data is near the end of the trace buffer. With MULTI 8, you no longer have to wait to access this data. With supported targets, MULTI automatically retrieves and decodes trace data backward so that you can analyze the most recent data quickly with History and TimeMachine. Additionally, backward trace retrieval results in much faster trace decoding and lower disk usage. For more information, see “Retrieving Trace Data Backward” in Chapter 20, “Analyzing Trace Data with the TimeMachine Tool Suite” in *MULTI: Debugging*.

Freeze-Mode Debug Snapshots

Previously, you could only save debug snapshots while debugging in run mode, but MULTI 8 now supports debug snapshots in freeze mode as well. Additionally, the new **debugsnapshot -traceonly** option allows you to exclude target memory and register values from a freeze-mode debug snapshot in order to generate the **.dsnap** more quickly. For more information, see Chapter 24, “Debug Snapshots” in *MULTI: Debugging*.

Debugging Support for C11, C++11, C++14, and C++17 Features

MULTI 8 now supports a number of new features, including:

- C++11 lambda functions
- C++14 variable templates
- Variables with thread-local storage in u-velOSity and INTEGRITY user space
- `char16_t`, `char32_t`, and `wchar_t` base types

To enable these features, you must build with Compiler 2022.1 or newer and use the following compiler options:

- **--dbo_version=40**
- **-dblink.args=-dnmVersion=43**

The required options may change if you are using a compiler version newer than 2022.1, so you should check the compiler release notes for updated product compatibility. Programs that are built with these options cannot be debugged with older versions of the IDE. Attempting to do so will result in errors such as the following:

```
multi: error: <libst>: "a.dnm" has version 43, multi only
supports up to version 41
multi: error: <libst>: "a.dla" has version 40, multi only
supports up to version 37
```

Note

The Debugger cannot debug `thread_local` variables in kernel space. If you attempt to display them in the Debugger, it will falsely report that they are optimized away.

History Window Improvements

Stateful Display for EAGLE-Logged Data in the History Window

History now gives you the ability to display data for EAGLE-logged variables as states associated with ranges of time. The stateful display implies that each logged value or string represents the variable up until the time another value or string was logged. This display may be useful if you are zoomed in on a range of time when no values or strings were logged, but when you want to see the value or string that was most recently logged. For more information, see “Displaying EAGLE Data as States or Impulses” in Chapter 23, “Viewing System Data with History” in *MULTI: Debugging*.

Improved Display of Signals for EAGLE-Logged Numerical Variables

A new coloring scheme in signals where numerical data is displayed as text shows relationships among values, allowing you to quickly spot high-level trends in logged data when you are zoomed out. For more information, see "Displaying Numerical Data as Text" in Chapter 23, "Viewing System Data with History" in *MULTI: Debugging*.

Additional Detailed Information About Signals

The History window now makes it even easier to gather information about your system at a glance. Density graphs display over certain signals to give you a visual representation of the frequency of events that occurred. Tooltips in the legend can display entry count, rate of entries, and minimum and maximum values for EAGLE-logged numerical variables. For more information, see "Density Graphs" in Chapter 23, "Viewing System Data with History" in *MULTI: Debugging* and "Tooltips" in Chapter 23, "Viewing System Data with History" in *MULTI: Debugging*.

Signal Groups for EAGLE and Task Signals

You can further customize the way information is displayed in History by assigning signal groups to EAGLE and task signals. Using signal groups, you can analyze composite views of specific signals within an address space in the graph pane. For more information, see "Signal Groups" in Chapter 23, "Viewing System Data with History" in *MULTI: Debugging*.

Support for Setting a Custom Relative Time Origin

New options to the **View** menu allow you to change the origin point of the time axis when displaying system time in History. For more information, see "The Time Axis" in Chapter 23, "Viewing System Data with History" in *MULTI: Debugging*.

Trace Data Supported in the History Window

The History window has replaced the PathAnalyzer in MULTI 8, allowing you to view all your system data in one place. This means you can now use all the features of History to analyze trace data you have collected. Additionally, History no longer requires instrumentation to display function calls, establish run-mode connections, or enable synchronous debugging. For more information, see “Viewing Trace Data with History” in Chapter 23, “Viewing System Data with History” in *MULTI: Debugging*.

Python 3 Support

The MULTI-Python integrated system now supports Python 3, but it is disabled by default. To enable Python 3 support, do one of the following:

- Set the environment variable `GHS_IDE_USE_PYTHON3` to a non-zero value. Setting it to `0` will disable Python 3.
- Store `2` or `3` in the **python_version.cfg** file for Python 2 or Python 3, respectively. MULTI searches for **python_version.cfg** in the following order:
  1. Your personal configuration directory (see “Your Personal Configuration Directory” in Chapter 7, “Configuring and Customizing MULTI” in *MULTI: Managing Projects and Configuring the IDE*)
  2. The **config** directory of the IDE installation
  3. The **defaults** directory of the IDE installation

  When the environment variable `GHS_IDE_USE_PYTHON3` is defined (regardless of the value), MULTI does not search for the **python_version.cfg** file.

Python 3 is compatible with Compiler 2021.1 and INTEGRITY 19 versions and later. If Python 3 support is enabled with older compiler or RTOS versions, certain IDE actions may fail with an error message such as `inconsistent use of tabs and spaces in indentation`.

If you are using a probe setup script, you will need to convert it to support Python 3. To do so, you can use a third-party tool, such as **expand**, to convert tabs to spaces in Python scripts. For example:

```
expand -i -t8 test.py > test_expanded.py
```

To convert Python 2 syntax into Python 3 syntax, you can use the Python 3 **2to3** utility.

Any-Task Breakpoints Supported in TimeMachine

Any-task breakpoints are now the default breakpoint type in TimeMachine. For more information, see "OSA Breakpoints in TimeMachine" in Chapter 20, "Analyzing Trace Data with the TimeMachine Tool Suite" in *MULTI: Debugging*.

Major Performance Improvements Using the Register View Window

Several changes have been made to the Register View window that drastically speed up the debugging experience. By default, the Register View window will only refresh the register values that are currently visible in the register tree. As you scroll, MULTI will refresh the new values that become visible. This allows for rapid run-control operations even when the context has a large number of registers.

To refresh all register values regardless of whether or not they are visible in the window, you can use the **wait -refresh** command or the **backgroundRefreshNonVisibleDataEntries** configuration option. Note that changing the default behavior will impact debugging operations.

Additionally, MULTI stores less information in Register View configuration files, which allows the Register View window to open more quickly. For more information about these changes, see the following release notes:

- "New Register View Window Configuration File" on page 20
- "Register Groups Only Visible on Certain Tabs" on page 21

Increased Number of Supported Registers

MULTI 8 now supports over 64,000 non-hardware registers (including register synonyms), as well as memory-mapped, dynamic, and MULTI command-based registers. Note that the maximum number of hardware registers, which can be accessed directly via a debug server, is still 64,000.

Host-Side Strings Supported in MULTI Expression Evaluation

All string constants issued from the MULTI Debugger command pane or **.rc** files are stored on the host. However, when they are assigned a program variable or used from the command line, MULTI will automatically save them on the target. For more information, see “Host-Side Strings in Expression Evaluation” in Chapter 14, “Using Expressions, Variables, and Function Calls” in *MULTI: Debugging*.

Target List Filtering

If your system contains many entries in the Target List, you can now filter it to display certain entries of interest. Using the new filter field, you can specify a string and the target list will display entries that include the string while hiding all others. For more information, see “Filtering Target List Entries” in Chapter 2, “The Main Debugger Window” in *MULTI: Debugging*.

Ability to Quickly Save Different Views of a Debug Snapshot

MULTI 8 allows you to save different views of a loaded debug snapshot. This can be useful if, for example, you are using one debug snapshot to debug multiple problems, or you and another engineer are working from the same debug snapshot, or you want to share a sequence of debugging steps with a colleague. Unlike debug snapshots, which include target memory and host-side information, debug snapshot views include only host-side information, such as History's state and the location of the INTEGRITY TimeMachine cursor. As a result, debug snapshot views are significantly smaller, making them much faster to save and easier to transfer than debug snapshots. For more information, see “Saving a Debug Snapshot View” in Chapter 24, “Debug Snapshots” in *MULTI: Debugging*.

Path Remapping for INDRT2 (rtserv2) Connections

The **rtserv2** debug server now allows you to remap a pathname on the target to an executable pathname on your host. This is especially useful if the program you want to debug on your INDRT2 target has the exact same executable pathname as an unrelated program on the MULTI Debugger host but you cannot rename either program. For more information, see the **rtserv2 -remap_path** option and the

remappath command in Chapter 4, “INDRT2 (rtserv2) Connections” in *MULTI: Debugging*.

Profile Window Improvements

The Profile window can now process and display coverage data up to four times faster than it did in MULTI 7. The display view has been enhanced to summarize additional profiling data from recordings. For example, you can now select the **Source** and **Assembly** tabs to browse data by source line and assembly instruction. The **Filter** tab also provides the option to exclude source lines and assembly addresses that account for less than 1% of the time from the Profile window and Debugger. For more information, see “The Display View” in Chapter 17, “Recording and Viewing Profiling Data” in *MULTI: Debugging*.

Improved 64-Bit Linux Native Debugging

With MULTI 8 and Compiler 2018.1 and newer, the Linux native debug server, **linuxserv**, automatically detects whether the program you are debugging is 32-bit or 64-bit. You can debug 32-bit and 64-bit executables in the same MULTI session with the same **linuxserv** debug server connection. As a result, the **linuxserv** debug server now displays two CPUs in the MULTI target list: one for 32-bit and one for 64-bit.

Support for Physical Memory Access

MULTI now supports physical memory access through a number of memory commands and corresponding menu items. For more information, see Chapter 10, “Memory Command Reference” in *MULTI: Debugging Command Reference*.

Memory Protection for Speculative Accesses

The MULTI Debugger uses a heuristic to perform automatic coherency checking and to walk the call stack. As a result, it may perform memory accesses that cause the target to crash. In an effort to prevent this, MULTI 8 defaults to marking these types of accesses as speculative. This information may then be used by the debug server to identify and prevent unsafe memory accesses. Note that not all debug servers support this capability and that this feature is only implemented for automatic coherency checking and call stack walking. It has no effect if memory is accessed in other ways (for example, if you issue the Debugger command **memview** ***address_expression***).

You can control this behavior with the new **useSpeculativeMemoryAccess** configuration option. For more information, see “Other Debugger Configuration Options” in Chapter 8, “Configuration Options” in *MULTI: Managing Projects and Configuring the IDE*.

Pointer Expressions Column in the Memory View Window

The Memory View window has a new **Pointer Expressions** column that simplifies manual call stack walking by displaying matching symbols for memory values. For more information, see the description in “The Column Submenu” in Chapter 37, “Memory View GUI Reference” in *MULTI: Debugging*.

Improved ROM Debugging

With the **memmark** command, you can specify which memory regions the MULTI Debugger should treat as ROM. Rather than having to manually specify these regions in the linker directives file, you can add **memmark** to a setup script so that it executes whenever the script is run, regardless of the program being debugged. Additionally, the **Run To This Line** menu option (**runtohere** command) is now supported when debugging in ROM. For more information, see Chapter 10, “Memory Command Reference” in *MULTI: Debugging Command Reference* and “Debugging a ROM Program” in Chapter 27, “Working with ROM” in *MULTI: Debugging*.

Real-Time and Arena malloc Support in Memory Allocations Window

When using Green Hills Compiler 2019.1 or newer, MULTI 8 supports heap walking with the **Real-Time** and **Arena** `malloc` implementations. Additionally, the **Visualization** tab of the Memory Allocations window displays even more information to help you better understand your program's heap and memory usage. For more information, see Chapter 16, “Viewing Memory Allocation Information” in *MULTI: Debugging*.

Additional Call Stack Information Supported in Memory Allocations Window

Previously, the **Leaks** and **Allocations** tabs of the Memory Allocations window supported showing information about more than five levels of the call stack only if you were using an x86-based Linux native target. In MULTI 8, the Memory Allocations window extends support for additional levels of call stack information to any target using a version of the Green Hills toolchain that supports this.

Ability to Configure Memory Cache Usage

It is now possible to control whether or not MULTI uses memory cache for a given task while the target is running. For more information, see `_CACHE_WHILE_RUNNING` in “System Variables” in Chapter 14, “Using Expressions, Variables, and Function Calls” in *MULTI: Debugging*.

Similarly, you can specify the maximum amount of target memory MULTI can use before cleaning up the cache. Doing so prevents MULTI from using too much memory and quitting unexpectedly. For more information, see **memoryCacheLimit** in “Other Debugger Configuration Options” in Chapter 8, “Configuration Options” in *MULTI: Managing Projects and Configuring the IDE*.

Support for Loading Multiple .mdv Files via Debug Information

In addition to using the **dvload** command, you can now load MULTI data visualization (**.mdv**) files with the `#pragma ghs debugstring "datavisu.foo filename"` directive. For more information, see “Loading and Clearing MULTI Data Visualization Files” in Appendix D, “Creating Custom Data Visualizations” in *MULTI: Debugging*.

Trace Timestamp Support on Simulators

When you use MULTI 8 with a Green Hills Compiler version 2020.1 and newer, trace timestamps are available on all simulators that support trace. This trace option, which is enabled by default, allows History to hide debug time and display local times. For more information, see “Configuring Simulator-Specific Trace Options” in Chapter 21, “Advanced Trace Configuration” in *MULTI: Debugging*.

Ability to Set Marks in the Editor

MULTI 8 gives you the ability to set a mark on any line of a file opened in the MULTI Editor. You can quickly jump to any line where you have set a mark. For more information, see "Using Marks" in Chapter 4, "Editing Files with the MULTI Editor" in *MULTI: Managing Projects and Configuring the IDE*.

Tab Visualization in the Editor

The new **charToVisualizeTabInIndentation** configuration option allows you to specify characters that represent tabs in the indentation of the MULTI Editor. Another new configuration option, **keepIndentationPrefix**, maintains the indentation from the previous line. For more information, see the descriptions of **charToVisualizeTabInIndentation** and **keepIndentationPrefix** in "Other MULTI Editor Options" in Chapter 8, "Configuration Options" in *MULTI: Managing Projects and Configuring the IDE*.

Wildcard Characters Supported on Windows

MULTI IDE programs now support wildcard expansion of filenames (using the `*` and `?` characters) on Windows hosts in accordance with standard Windows rules.

Subversion 1.9.x Support

Subversion 1.9.x is now supported with MULTI 8. For more information, see "Integrating with Subversion" in Chapter 5, "Integrating MULTI with a Version Control System" in *MULTI: Managing Projects and Configuring the IDE*.

Enhanced Support for High-DPI Displays

The MULTI IDE now provides improved support for running on high-DPI displays. MULTI uses the system DPI settings from the display settings in Windows or from the desktop environment, when available, in Linux. On Windows, the MULTI IDE supports system DPI awareness, which is based on the DPI of the primary monitor at login. For the most consistent behavior, we recommend that you adjust the scaling while the IDE is closed.

Full Color Configuration Available on Windows

Windows users can now load color schemes and configure the foreground, background, control, and selection colors separately from the Windows system colors. For more information, see “Colors Configuration Options” in Chapter 8, “Configuration Options” in *MULTI: Managing Projects and Configuring the IDE*.

Support for Anti-Aliased Fonts in Linux

Anti-aliased fonts are now supported on Linux systems that have the **libxft2** library installed. Older versions of Xvnc, such as RealVNC 4.0, may not support recent X server extensions, which may result in sluggish redraw in the MULTI IDE when using a remote X display. We recommend upgrading to a more modern Xvnc server. Otherwise, you can disable the use of anti-aliased fonts by setting the environment variable `GHS_DISABLE_XFT` to `1`.

Local License Server (LLS)

A Local License Server (LLS) process is launched for clients sharing the same user name, host, display host, display ID, and `GHS_LMHOST` setting. This means that multiple LLS processes can be launched in different environments for the same user on the same host with the same display ID but *different* `GHS_LMHOST` settings. For more information, see “Local License Server (LLS)” in Chapter 2, “Managing Floating and Named-User Licenses” in *MULTI: Licensing*.

Command to Display Global Symbols Using Wildcards

With the new **symfind** command, you can search field paths from global or static variables in a debugged application using patterns that contain wildcards. For more information, see “Using symfind to Show Global Data Symbols” in Chapter 14, “Using Expressions, Variables, and Function Calls” in *MULTI: Debugging*.

New Options to the addhook Command

With MULTI 8, you can add Debugger hooks to execute commands after the target powers on or off by using the **addhook** command with the new **-after poweron** and **-after poweroff** arguments. If MULTI detects that the target has powered off, it will automatically update the Debugger's **Status** Column. For more information, see “Hook Commands” in Chapter 15, “Scripting Command Reference” in *MULTI: Debugging Command Reference*.

New Options to the debugbutton Command

New options to the **debugbutton** command allow you to control whether Debugger buttons that you create or configure appear to be pushed down or dimmed. For more information, see the descriptions of ***dim_var*** and ***pushed_var*** in “Configuring and Customizing Toolbar Buttons” in Chapter 7, “Configuring and Customizing MULTI” in *MULTI: Managing Projects and Configuring the IDE*.

New Options to the debugsnapshot Command

When you create a debug snapshot with the **debugsnapshot** command, you can use the **-files** option to add or remove specific files whenever a **.dsnap** file is generated. For more information, see “debugsnapshot” in Chapter 2, “General Debugger Command Reference” in *MULTI: Debugging Command Reference*.

New Option to the disassemble Command

The functionality of the **disassemble** command has been extended in two ways. The new **-opcodes** option causes the Debugger to output raw hex opcodes, along with assembly mnemonics, for each machine instruction. Previously, the **disassemble** command was only able to output assembly mnemonics. Also, the **disassemble** command will now attempt to identify known data-in-text regions and output them as data directives (for example, `.byte`) by default. For more information, see “disassemble” in Chapter 10, “Memory Command Reference” in *MULTI: Debugging Command Reference*.

New Option to the e Command

The functionality of the **e** command has been extended to allow for navigation to a specific symbol name, including all types, `enum` members, variables with constant values, and class/struct non-static fields. Previously, the **e** command could only navigate to a given address. For more information, see “e” in Chapter 11, “Navigation Command Reference” in *MULTI: Debugging Command Reference*.

New Option to the l Command

The **l** command now lists target connections when it is specified in conjunction with the new option **C**. For more information, see “l” in Chapter 8, “Display and Print Command Reference” in *MULTI: Debugging Command Reference*.

New Option to the new Command

A new option to the **new** command allows you to associate a program with a specific CPU. The **-core *core_number*** option prevents the appearance of an error such as `Component has more than one CPU` when more than one CPU is available. For more information, see “new” in Chapter 2, “General Debugger Command Reference” in *MULTI: Debugging Command Reference*.

New Option to the output Command

The **output** command now provides the option to display a timestamp before each message in an output log file. For more information about the **-timestamp** option, see “output” in Chapter 2, “General Debugger Command Reference” in *MULTI: Debugging Command Reference*.

New Option to the route Command

The new **use_default_context** option allows you to route commands to a context associated with a debug server component. (This option should only be used when a known thread context is not available.) For more information, see “route” in Chapter 15, “Scripting Command Reference” in *MULTI: Debugging Command Reference*.

New Option to the sourceroot Command

The **sourceroot remap -append** option adds specified directories to the end of any existing remappings previously created with **sourceroot remap**. For more information, see “sourceroot” in Chapter 6, “Configuration Command Reference” in *MULTI: Debugging Command Reference*.

New Options to the trace Command

The **trace retrieve** command now has several additional options that allow you to specify exactly how much trace data to collect from your probe or target. With the **loadtrigger** and **savetrigger** options, you can load and save trigger settings as you would in the Set Triggers window. The **status** option prints the current status of trace retrieval and decoding to the Debugger command pane. The **stats** option opens the Trace Statistics window, which displays even more information about your trace data. For more information about these and other **trace** options, see “trace” in Chapter 19, “TimeMachine and INTEGRITY TimeMachine Command Reference” in *MULTI: Debugging Command Reference*.

Changes in Behavior

Base Class Pointers Now Cast to Derived Class Type

The MULTI Debugger now attempts to cast base class pointers to the derived class type when printing dereferenced pointers in the command pane. You can control this behavior with the new **printDerived** configuration option. For more information, see “Other Debugger Configuration Options” in Chapter 8, “Configuration Options” in *MULTI: Managing Projects and Configuring the IDE*.

Memory Test Wizard Now Supports 64-Bit Addresses

The Memory Test Wizard now accepts 64-bit addresses in its **Start address** and **End address** fields. Previously, it interpreted addresses entered in these fields as signed 32-bit integers.

Issuing the flash burn Command No Longer Prompts a Dialog

In MULTI 7, issuing the **flash burn** command would open a download progress dialog that displayed output from the setup script. The **flash burn** command no longer prompts this dialog, and the setup script output is printed to the Debugger command pane instead.

MULTI Waits for Run-Mode Task Initialization with the wait Command

When you issue the **wait** command on a task from an **rtserv2** run-mode connection, MULTI will wait for it to initialize by default. This does not apply if the task status suggests that initialization may not happen (e.g. `DebugBrk` or `Incipient`). For more information, see the **-initialized** and **-no_initialized** options in “wait” in Chapter 2, “General Debugger Command Reference” in *MULTI: Debugging Command Reference*.

Error Message When filedialog Command is Canceled

Previously, the **filedialog** command would not return an error if the **Cancel** button was selected in the file chooser. The **filedialog** command will now abort the script and return an error if the **Cancel** button is selected. To prevent such errors from halting script execution, see the **Continue running script files on error** configuration option in *MULTI: Managing Projects and Configuring the IDE*.

Ctrl+Shift+1 Now Bound to UnMark Command in Editor

The Editor keyboard shortcut **Ctrl**+**Shift**+**1** (that is a number one, not a lowercase `L`) was previously bound to the Editor command **AllowAutoCheckout**. In MULTI 8, **Ctrl**+**Shift**+**1** is bound to the Editor command **UnMark**. The old binding was not documented.

Lines Containing ghs_noprofile Highlighted Differently in the Debugger

Previously, lines containing functions marked with the `ghs_noprofile` function attribute and the code paths that call these functions were highlighted in a single color, regardless of whether they were covered or not. Uncovered lines are now highlighted in beige, and covered or partially covered lines are highlighted in light green, by default. For more information about configuring these colors, see the **diffHighlight** and **noprofileHighlight** options in “The More Color Options Dialog” in Chapter 8, “Configuration Options” in *MULTI: Managing Projects and Configuring the IDE*.

Debugger Support for Nested Functions

MULTI 8 provides support for nested functions, such as C++11 lambda functions. Breakdots and function-relative numbers now display properly in the Debugger in both source-only mode and interlaced assembly mode. For more information, see “Function-Relative Line Numbers and Nested Functions” in Chapter 2, “The Main Debugger Window” in *MULTI: Debugging*.

New wait Command Behavior May Require Changes to User Scripts

Previously, the **wait** command would return immediately, but further commands were not processed until the **wait** condition was satisfied. In MULTI 8, the **wait** command now returns after the **wait** condition is satisfied. This change can affect certain behaviors, such as where output is sent or the order of operations between MULTI and the Python scripting interface.

For example, in earlier versions of MULTI, if you used the Python interface to send a **wait** command to MULTI with the `RunCommands` Python function, the output generated while waiting for the **wait** condition was not always sent back to Python. In MULTI 8, this output is correctly collected and returned to Python. If you need assistance porting scripts to MULTI 8, please contact Green Hills Support.

For more information about the `RunCommands` function, see “RunCommands()” in Chapter 7, “Window Functions” in *MULTI: Scripting*.

New Register View Window Configuration File

The Register View window has a new configuration file that affects the expansion state of register groups. Previously, when you closed a Register View window, it would remember whether each register group was collapsed upon subsequent launches. In MULTI 8, the Register View window only remembers whether visible groups and registers in the register hierarchy are collapsed. This change allows the Register View window to open much more quickly.

For more information, see “Register View Window Configuration Files” in Chapter 13, “Viewing and Customizing Registers” in *MULTI: Debugging*.

Register Groups Only Visible on Certain Tabs

Selecting the **View → Show Hidden Registers** option in the Register View window no longer results in displaying all registers on every tab. In some cases (listed below), certain registers will not display on every tab, even if this option is selected. This results in far less entries in the Register View configuration file, which improves performance when launching the Register View window with a large number of registers.

- If a register is bound to a specific tab, it will only display in that tab.
- A user-defined tab will only display registers (and the related groups) that are explicitly bound to it.
- If a register is not bound to a specific tab, it will only display in the default tab (**Processor** or **Board**) based on its access method.

Register View Tabs Displayed if Register Definition Files Are Loaded

In previous versions of MULTI, loading a register definition file ensured that the tabs referred to in the file would be displayed upon all subsequent launches of the Register View window (assuming you were debugging the same target or program). This was true even if the register definition file was no longer loaded. Additionally, tabs you created via the **regtab -insert** command or the Register View window's **View → Add New Tab** menu option always appeared in the Register View window (assuming the same target or program).

In MULTI 8, tabs—including those created via the above methods—that are referred to in a register definition file are not displayed in the Register View window unless the register definition file is loaded.

Windows Restored When Loading a Debug Snapshot

Before MULTI 8, the Memory View, Breakpoints, and Note Browser windows were available in a debug snapshot, but not restored. The settings for those windows are now saved and restored when you load a debug snapshot.

Call Browser Replaced by the Search Window

You can now browse calls to a function in History using the Search window. Previously, if you wanted to view every logged call to a function, History would open a Call Browser, which is removed in MULTI 8. History will now open the Search window and perform a **Single Events Function** search, allowing you to view and compare all your searches in one place. For more information, see "Browsing Calls to a Function" in Chapter 23, "Viewing System Data with History" in *MULTI: Debugging*.

print Command Outputs C and C++ Bitfields of char Type as Integers

Previously, the **print** command displayed the values of C and C++ bitfields of `char` type as characters. In MULTI 8, the **print** command displays these values as integers.

Command Contexts Preserved Across Symbol Merging

MULTI will not necessarily delete commands when merging symbol and hardware contexts. If there are commands on the hardware context but not on the symbol context, MULTI will merge those commands onto the symbol context before destroying the hardware context. The Debugger will display a warning message if commands are thrown away due to command failure or commands are unintentionally deleted from a context that was destroyed. You can disable this message with the new **warnWhenLoseCommandFromInputStack** configuration option. For more information, see "Command Contexts" in Chapter 2, "The Main Debugger Window" in *MULTI: Debugging*.

TimeMachine Changes with SMP Targets

If a core is already in TimeMachine mode, clicking the TimeMachine button will always switch the selected task into TimeMachine mode on that core. When choosing which core to put in TimeMachine mode, MULTI no longer considers whether the task is currently executing on a core. Instead, MULTI looks at the trace data to determine which core was most recently traced executing the task.

Opcode Information Provided Before Trace Decoding

Previously, MULTI would try to read opcodes from the target as needed during trace decoding, depending on the **Abort processing on opcode failure** and **Read unknown opcodes from target (may halt target)** options in the Trace Options window. Those options have been removed in MULTI 8. Instead, you can provide opcodes that are not in a loaded ELF file by specifying the new **add_text_range**, **del_text_range**, and **print_text_ranges** options to the **trace** command. For more information, see "trace" in Chapter 19, "TimeMachine and INTEGRITY TimeMachine Command Reference" in *MULTI: Debugging Command Reference*.

For additional information, see "Solving Trace Decoding Problems Caused by Missing Opcodes" in Chapter 20, "Analyzing Trace Data with the TimeMachine Tool Suite" in *MULTI: Debugging*.

Previously Retrieved Trace Discarded When Retrieving Trace Backward

By default, MULTI now retrieves and decodes trace data backward (on supported targets). When retrieving trace data backward, all previously decoded data is discarded, and the whole trace buffer is retrieved and decoded again. In contrast, when trace is decoded forward, previously retrieved and decoded data is retained. Newly collected data is retrieved and decoded, and the newly decoded data is appended to the previously decoded data. For more information, see "Retrieving Trace Data Backward" in Chapter 20, "Analyzing Trace Data with the TimeMachine Tool Suite" in *MULTI: Debugging*.

Trace Processing Aborted When Disk Space is Low

Before MULTI 8, if your host disk ran out of space while collecting trace data, MULTI would delete trace data based on age, triggers, bookmarks, and current selection. However, this had adverse effects such as gaps in timestamps displayed in the Trace List. To avoid such issues in this release, MULTI detects when the host disk is running out of space, then aborts trace collection and displays an error message, saving any trace data that has already been decoded. As a result of these changes, the **Host buffer size** option has been removed from the Trace Options window. For more information, see "Trace Disk Usage" in Chapter 20, "Analyzing Trace Data with the TimeMachine Tool Suite" in *MULTI: Debugging*.

Trace Sessions Replaced with Debug Snapshots

Note that the ability to save or load a debug snapshot is a separately licensed feature. Trace sessions (**.trs** files) are deprecated in MULTI 8. The menu options for saving and loading trace sessions have been removed; the corresponding commands **tracesave** and **traceload** still exist in this release but with different behaviors (see "tracesave and traceload Functionality" on page 24). Debug snapshots can do much more than trace sessions, like the following:

- Saving live target state
- Saving Debugger and History state
- Saving source
- Generating smaller files due to compression

For more information, see "Freeze-Mode Debug Snapshots" in Chapter 24, "Debug Snapshots" in *MULTI: Debugging*.

tracesave and traceload Functionality

The **tracesave** and **traceload** commands are deprecated in MULTI 8; however, they still exist but with different behaviors. The **tracesave** command now saves a debug snapshot instead of a **.trs** file. The **traceload** command can now load either a **.trs** file or a **.dsnap** file.

Trace Options Can Be Set in the Command Pane

You can now configure the settings in the Trace Options window using the **trace set** command. Previously, you could only set target-specific trace options with this command. For more information, see "trace" in Chapter 19, "TimeMachine and INTEGRITY TimeMachine Command Reference" in *MULTI: Debugging Command Reference*.

Trace Triggers No Longer Automatically Bookmarked

Trace triggers are no longer automatically bookmarked in MULTI 8, but you can still see when trace triggers were hit by viewing the **Trace Trigger** signal in History. For most trace architectures, a trigger only goes off one time after it is armed.

However, for some architectures, another trigger packet is emitted each time the trigger condition is met. In that case, you can get a list of all trigger events by right-clicking the **Trace Trigger** signal in the legend and selecting **Search String**. For more information, see “Viewing Trace Data with History” in Chapter 23, “Viewing System Data with History” in *MULTI: Debugging*.

Hardware Breakpoints No Longer Synchronized Between TimeMachine and INTEGRITY Targets

If you are debugging code running on INTEGRITY and you enter TimeMachine mode, MULTI will no longer apply hardware breakpoints to TimeMachine that were set on the target. Similarly, when you leave TimeMachine mode, MULTI will not apply hardware breakpoints to the target that were set in TimeMachine.

Warning Message Removed For Shadowed MULTI Variables

MULTI no longer displays the often confusing warning message for MULTI variables whose names exist (without the prefix $) in the program being debugged.

MULTI Searches for First Matching Data Description in .mdv Files

Previously, when searching for a data description to use for a variable, MULTI would start at the bottom of the **.mdv** file and use the last matching description. MULTI 8 starts at the top of the **.mdv** file instead and uses the first matching data description. In rare cases this could require changes to **.mdv** files. If this is a problem, contact Green Hills Support.

Improvements to the Graphical Probe Administrator

The Graphical Probe Administrator (**gpadmin**) no longer displays the Probe List window, which caused issues in previous releases. You can still use **gpadmin** to configure and update firmware for your Green Hills probes. Additionally, USB connections from **gpadmin** are now supported for Linux. For more information, see “The Graphical Probe Administrator” in Chapter 1, “Administering Your Probe” in *Green Hills Debug Probes User's Guide*.

DWARF Debugging Information Automatically Translated

If you have licensed the **dwarf2dbo** utility, MULTI now automatically translates DWARF debugging information generated from executables built by third-party compilers. Previously, this had to be done manually using the **autoDwarf2Dbo** configuration option, which no longer exists. For more information, see “Using the Debugger with Third-Party Compilers” in Appendix C, “Using Third-Party Tools with the MULTI Debugger” in *MULTI: Debugging*.

Non-GUI Mode Support for the osatasklist Command

MULTI 8 provides added support for the **osatasklist** command, which can now be used in both non-GUI and GUI modes.

lic_find_licenses Utility Prints Exact License Use Time

Specifying the **-w** option with the **lic_find_licenses** utility now outputs exactly how long a license has been checked out rather than printing truncated data. For example, it will now print `10d2h0m10s` instead of `10d2h`. For more information, see “Output Format with -w” in Chapter 4, “Viewing License Information” in *MULTI: Licensing*.

Deprecated Features

The following features are deprecated in MULTI 8:

- PathAnalyzer (replaced with History)
- EventAnalyzer trace integration (only accessible with the **tracemevsys** command)
- **tracepath** command, which opens PathAnalyzer
- **tracesave** and **traceload** commands
- Trace session (**.trs**) files
- TimeMachine API file interface
- Support for the at sign (@) wildcard symbol
- Windows 7 support
- INTEGRITY 5 and 10 support

- Tracepoints and passive mode (this includes the tracepoint-related commands listed in Appendix A, “Deprecated Command Reference” in *MULTI: Debugging Command Reference*)
- Support for ClearCase version control system

Removed Features

The following features are not available in MULTI 8:

- Solaris support
- Support for 32-bit hosts
- SourceSafe integration
- **stabs2dbo** utility
- **paddedHex** configuration option (use **viewDef PadHex** instead)
- **disAsmStyle** configuration option
- `ShowAddress`, `ShowFType`, `ShowAllFields`, `ExpandValue`, `ExpandComplexMemberValue`, `OpenPointer`, and `ShowChanges` settings of **viewDef** configuration option
- **--ghslm** and **--legacy** command line options to the **servecode** utility (legacy licensing technology, aka Elan, is no longer supported)
- `GHS_LMPORT` configuration variable. Use the `GHS_LMHOST` variable to specify TCP ports instead (see “Configuration Variables for Client Machines” in Chapter 3, “Configuring End User Machines” in *MULTI: Licensing*).

If your applications or systems rely on any of the preceding features, please contact your Green Hills sales representative.

Chapter 2

Changes in MULTI 7

Contents

New Features and Enhancements 30
Changes in Behavior 39
Deprecated Features 44
Removed Features 45

New Features and Enhancements

History

MULTI 7 includes a new debugging and performance analysis tool called History, which provides a visual, time-based overview of your system. You can use History to:

- Fix bugs
- Improve performance
- Gain a better understanding of your system

For case studies, see "Solving Performance Problems" in Appendix A, "Case Studies" in *MULTI: Debugging Techniques and Performance Analysis*.

INTEGRITY TimeMachine

MULTI 7 introduces the INTEGRITY TimeMachine Debugger, which, like the TimeMachine Debugger, allows you to debug backwards in your code. The new INTEGRITY TimeMachine Debugger:

- Provides full OS integration, with awareness of kernel events and shared memory
- Works with run-mode connections to give you a smoother and faster debugging experience
- Instantaneously transitions into INTEGRITY TimeMachine mode when you step or run back
- Does not depend on hardware trace data, which means that data values are always available, data is never lost, and you are not limited to using CPUs and boards with trace capabilities

For more information, see Chapter 19, "Debugging with INTEGRITY TimeMachine" in *MULTI: Debugging*.

Debug Snapshot

MULTI 7 gives you the ability to save a debug snapshot to a file on your host machine and view the saved snapshot in the Debugger without a target connection. This is especially useful if you want to share a debugging session with another engineer, or if you need to repurpose your hardware before you are finished debugging a problem.

For more information, see Chapter 24, "Debug Snapshots" in *MULTI: Debugging*.

Synchronous Run Control

With a run-mode connection to INTEGRITY IoT, halting one task for debugging purposes causes other tasks in the system to stop in a non-invasive pause. Stepping a halted task or resuming the system causes paused tasks to become schedulable again. This means that interactions between tasks are minimally impacted by debugging. For more information, see "Stopping and Starting Tasks Synchronously" in Chapter 8, "Executing and Controlling Your Program from the Debugger" in *MULTI: Debugging*.

With a supported freeze-mode connection to a multi-core hardware target, all cores are run or halted at the same time. Halting all cores simultaneously means that you can debug one core without worrying that operations running on another core will affect shared memory. For more information, see "Synchronous Debugging on a Multi-Core Target" in Chapter 25, "Freeze-Mode Debugging and OS-Awareness" in *MULTI: Debugging*.

Profile Window

The **Profile** window and many profiling features have been redesigned to make it easier than ever to:

- Find performance problems
- Find coverage gaps and track test coverage of specific code
- Share profile data with other users
- Collect and merge profile data from several tests or profile recordings

For more information, see Chapter 17, “Recording and Viewing Profiling Data” in *MULTI: Debugging*.

Project Build Configurations

Projects can now define multiple build configurations. Each of these configurations sets different Builder and driver options so that you can build one project in different ways. For example, a debug configuration might set options that configure a project for maximum debugging ability, while a release configuration would optimize for speed and size. Project build configurations are supported with Compiler 2015.1 and newer. For more information, see “Displaying Project Build Configurations” in Chapter 2, “Managing and Building Projects with the Project Manager” in *MULTI: Managing Projects and Configuring the IDE*.

INTEGRITY Monolith Debugging

Because many INTEGRITY systems in final deployment incorporate a monolith application, MULTI 7 contains special support for run-mode debugging of INTEGRITY monolith applications. When you open an INTEGRITY monolith in the Debugger, the run-mode virtual AddressSpaces and Tasks defined statically in that monolith are shown in the Debugger even before the kernel has booted, and a subset of run-mode debugging features are available. For more information, see “Working with INTEGRITY Monolith Applications” in Chapter 22, “Run-Mode Debugging” in *MULTI: Debugging*.

Simplified Multi-Core Configuration

The MULTI 7 Debugger can use information you provide in a multi-core configuration file (***.ghsmc**) to automatically download one or more executables to your multi-core target and load symbol files in the Debugger. This simplifies target setup considerably. For more information, see “Using a Multi-Core Configuration File to Load Executables onto a Multi-Core Target” in Chapter 25, “Freeze-Mode Debugging and OS-Awareness” in *MULTI: Debugging*.

Automatic Setup of Certain Multi-Core Targets

If you are running an INTEGRITY application on an SMP multi-core target with the new synchronous debugging feature enabled, you can debug the application just like you would on a single-core target: the Debugger automatically runs the target setup script (if any), downloads the application on the first core of your multi-core target, and makes the application's symbol information visible on all cores. For more information, see “Synchronous Debugging of an INTEGRITY Application on an SMP Target” in Chapter 25, “Freeze-Mode Debugging and OS-Awareness” in *MULTI: Debugging*.

Choose Which Target List Entries are Displayed

You can now choose which entries are displayed in the target list. Click the star alongside each entry of interest, and then click the **Only show starred items** button (★), located to the right of the **Status** column. This button toggles the exclusive display of starred entries, their ancestors, the currently selected target list entry, and stopped entries. This feature is especially useful when your system has a large number of target list entries, but you only want to view a small number of them. For more information, see “The Star Column” in Chapter 2, “The Main Debugger Window” in *MULTI: Debugging*.

Choose Which Data Explorer Entries are Displayed

You can now choose which entries are displayed in the Data Explorer. Click the star alongside each item of interest, and then click the **Only show starred items** button (★) to toggle the exclusive display of starred items and their ancestors and descendants. This feature is especially useful if you are debugging an application that involves data structures with many fields, or complex functions with many local variables. For more information, see “Filtering Data Explorer Variables” in Chapter 11, “Viewing and Modifying Variables with the Data Explorer” in *MULTI: Debugging*.

Run to the Next/Previous Modification of a Variable

In MULTI 7, you can run to the next or previous modification of a variable from the Debugger source pane, the Data Explorer, or the command pane. For more information, see the descriptions of **Run to Next Modification** and **Run Back to Previous Modification** in “The Tools Menu” in Chapter 32, “Data Explorer GUI Reference” in *MULTI: Debugging*. Or see the same descriptions in “The Variable Shortcut Menu” in Chapter 33, “Debugger GUI Reference” in *MULTI: Debugging*.

Set Watchpoints from the Data Explorer

The new **WP** column in the Data Explorer allows you to set, toggle, and delete watchpoints—hardware breakpoints set on a variable's address that stop your program when the address is modified. For more information, see “Setting Watchpoints on Data Explorer Variables” in Chapter 11, “Viewing and Modifying Variables with the Data Explorer” in *MULTI: Debugging*.

Quickly Navigate Complex Data Structures in the Data Explorer

Context-sensitive arrows in the **Variable** column of the Data Explorer now allow you to quickly navigate complex data structures. Right-facing arrows reroot on child nodes, and left-facing arrows reroot on parent nodes. For more information, see “Navigating Complex Data Structures” in Chapter 11, “Viewing and Modifying Variables with the Data Explorer” in *MULTI: Debugging*.

Data Explorer Column Settings Remembered Within and Across Sessions

The Data Explorer now remembers column visibility and ordering within and across sessions.

Commands to Time Target Operations

You can use the new **starttime** and **endtime** commands to quickly get an estimate of how long a target operation takes. For more information, see the descriptions of **starttime** and **endtime** in Chapter 2, “General Debugger Command Reference” in *MULTI: Debugging Command Reference*. For an example use case, see “Measuring the Time Between Target Events” in Chapter 4, “Tips and Tricks” in *MULTI: Debugging Techniques and Performance Analysis*.

Command to Control Breakpoint Installation After Target Setup

You can use the new **install_bp_on_request** command in setup scripts to control precisely when breakpoints are installed. This is useful if, during the initialization of your target, you need to execute code on the target while it is not safe to install breakpoints. For more information, see "install_bp_on_request" in Chapter 3, "Breakpoint Command Reference" in *MULTI: Debugging Command Reference*.

Command to List and Optionally Attach to OSA Tasks

Like the **osaexplorer** command, the new **osatasklist** command provides information about OSA tasks; however, it typically accesses less target memory than **osaexplorer**. Additionally, it allows you to debug tasks in freeze mode. For more information about this command, see "osatasklist" in Chapter 15, "Scripting Command Reference" in *MULTI: Debugging Command Reference*.

Command to Clean up OSA Information or Remove OSA Support

The new **osacleanup** command cleans up OSA information or removes OSA support. This lessens or eliminates the effect of the OSA package on system performance, thereby speeding up operations like single-stepping. For more information, see "osacleanup" in Chapter 15, "Scripting Command Reference" in *MULTI: Debugging Command Reference*.

Command to Modify Target-Specific Mode Variables

The new **asmmode** command changes target-specific mode variables, which may affect the behavior of the disassembler and/or other target-specific behavior in the Debugger. The primary use for this is to set a specific instruction set mode for disassembly (such as Thumb when using ARM). Normally MULTI determines this from debug information, so it should only be necessary when looking at instructions that have no debug information. For more information, see "asmmode" in Chapter 8, "Display and Print Command Reference" in *MULTI: Debugging Command Reference*.

Added Options to the wait Command for Scripting

Several new options are available for use with the **wait** command, which blocks command processing until a specified action has occurred. New options are:

- `-timeoutcmd {`*`commands`*`}` — Executes *`commands`* if a timeout occurs before the specified action.
- `-pause` — Blocks command processing until the system is paused.
- `-debugsnapshot` — Blocks command processing until the debug snapshot save is complete.
- `-load` — Blocks command processing until all dynamic downloads for the current connection are complete.
- `-unload` — Blocks command processing until the last module to be unloaded is fully unloaded from the target.

For more information, see “wait” in Chapter 2, “General Debugger Command Reference” in *MULTI: Debugging Command Reference*.

Added Options to the load, unload, and prepare_target Commands

New options to the **load**, **unload**, and **prepare_target** commands allow you to block command processing until a load/unload operation is complete or until a specified timeout has occurred. For more information, see the documentation for **load**, **unload**, and **prepare_target** in “General Target Connection Commands” in Chapter 17, “Target Connection Command Reference” in *MULTI: Debugging Command Reference*.

Reduced Trace Disk Usage and Trace Session File Sizes

MULTI 7 dramatically reduces trace disk usage and trace session file sizes.

Tracing 64-bit INTEGRITY Targets Now Supported

MULTI 7 adds support for tracing 64-bit INTEGRITY targets.

Further Parallelization of Trace Processing

Trace processing has been further parallelized in MULTI 7. This results in a 20-30% speedup in trace processing for certain targets when you are decoding on a host PC with at least 4 cores.

Faster Reporting of Memory Leaks and Allocations

If you are using MULTI 7 and compiler 2014.1 or later, reporting of many memory leaks or allocations can be over 30 times faster than in previous releases.

Line Numbers Now Displayed in the Editor

By default, line numbers are now displayed in the Editor to make it easy to shift from looking at code in the Debugger to code in the Editor. For information about the configuration option that controls the appearance of line numbers—**Show line number**—see "The More Editor Options Dialog Box" in Chapter 8, "Configuration Options" in *MULTI: Managing Projects and Configuring the IDE*.

Display of Task Priority, Stack Usage, and Stack Size

If you are using INTEGRITY version 10 or later, the target list displays task priority, stack usage, or stack size when you right-click any column header in the target list and select **Priority**, **Stack used**, or **Stack size**, respectively. If you are using INTEGRITY IoT, **Priority** and **Stack size** are available.

Long Target List Entries Now Easy to Tell Apart

Compressed target list entries that do not fit in the **Target** column are now abbreviated so that you can easily tell entries apart and find what you are looking for. This is especially helpful when the target list is located on the left side of the Debugger window, where there is typically less room for it.

Improved Target List Display of OSA Tasks on Certain Multi-Core Targets

If you are running an INTEGRITY application on an SMP multi-core target with OS-aware debugging and the new synchronous debugging feature enabled, the Debugger's target list displays all OSA tasks in a single, system-wide task hierarchy. This differs from the display that is available when synchronous debugging is not enabled. In that case, a task hierarchy is nested under each core, so that if your multi-core target contains four cores, for example, each task appears in the target list four times. Not only is it easier to find and track tasks in the single, system-wide task hierarchy, the task statuses provided are always accurate as a consequence of the fact that all cores run and halt at the same time.

New Register access Type

A new register `access` type, `command`, allows you to define a register whose value is the output of a specified MULTI Debugger command. This is particularly useful in conjunction with the **mprintf** command. For more information, see the description of `access` in “The register Section” in Appendix E, “Register Definition and Configuration Reference” in *MULTI: Debugging*.

Python Interpreter Upgraded to 2.7.3

MULTI 7 contains an upgraded Python interpreter based on the 2.7.3 release of Python.

Subversion 1.7.x and 1.8.x Supported

MULTI 7 adds support for Subversion 1.7.x and 1.8.x.

Multiple TCP Ports Supported per License Client

The use of multiple TCP ports per license client is now supported. For more information, see “Specifying a Floating or Named-User License Pool” in Chapter 3, “Configuring End User Machines” in *MULTI: Licensing*.

Simpler License Manager Configuration

You can now use the **License Manager Settings** dialog box to configure a License Manager to serve licenses from multiple license files.

Changes in Behavior

Any-Task Breakpoints Set By Default in INTEGRITY

If you are using INTEGRITY or another operating system that supports any-task and task-specific breakpoints, any-task breakpoints are usually preferred, and they are now the default. For information about modifying this behavior, see the configuration option **clickToSetAnyTaskBp** in “Other Debugger Configuration Options” in Chapter 8, “Configuration Options” in *MULTI: Managing Projects and Configuring the IDE*. See also “Breakdots and Breakpoint Markers” in Chapter 8, “Executing and Controlling Your Program from the Debugger” in *MULTI: Debugging*.

Watchpoints Implemented Using Any-Task Hardware Breakpoints

Watchpoints are implemented using any-task hardware breakpoints on targets that support them. Because an any-task hardware breakpoint can be hit by any task in the address space in which it is set, you can capture writes to memory locations of interest regardless of which task the write comes from. Furthermore, you can capture writes even if the writing task did not exist when you set the watchpoint.

Target List Located on Left By Default

The target list is now located on the left side of the Debugger window by default. To move it to the top of the Debugger, click the **Move target list to top** button ( ), located to the right of the **Status** column.

New Methods for Manually Dumping and Processing Profiling Data

The methods for manually dumping and processing profiling data that were available in MULTI 6 are not supported in MULTI 7. This includes use of the **profdump** and **profilemode process** commands. For information about current profiling techniques, see Chapter 17, “Recording and Viewing Profiling Data” in *MULTI: Debugging*.

Several Profiling Commands Consolidated Under the `Profile' Command

The **profdump**, **profilemode**, and **profilereport** commands are no longer supported, and many of them are no longer necessary. The following table provides alternative commands where available.

Old Command	New Command
profdump	**profile stop**
profilemode clear	**profile clear**
profilemode close	**profile close**
profilemode coverage|percent	**profile display coverage|percent**
profilemode import	**profile load**
profilemode process	**profile collect**
profilemode start|stop	**profile start|stop**
profilereport print	**profile print**
profilereport save	**profile csvexport**

For more information, see Chapter 12, “Profiling Command Reference” in *MULTI: Debugging Command Reference* and Chapter 17, “Recording and Viewing Profiling Data” in *MULTI: Debugging*.

Continual Update of Profile Window No Longer Available

The **Profile** window no longer continually updates if you are profiling all tasks in the system or using trace. Profiling data can be recorded, or trace data can be converted into a profile recording, by using the on-demand mechanisms described in:

- “Recording Sampled and Instrumented Profiling Data” in Chapter 17, “Recording and Viewing Profiling Data” in *MULTI: Debugging*, and
- “Converting Trace Data into a Recording” in Chapter 17, “Recording and Viewing Profiling Data” in *MULTI: Debugging*, respectively.

If you are using trace and have enabled the option **Retrieve trace when buffer fills**, you must first process trace profiling once at the beginning of your session in order to start the processing of trace profiling. Once you have run the target as desired, you can process trace profiling again to obtain the final results. For information about processing trace profiling, see “Converting Trace Data into a Recording” in Chapter 17, “Recording and Viewing Profiling Data” in *MULTI: Debugging*.

Data Explorer No Longer Treats Instance Variables as Local Variables

In previous versions of MULTI, issuing the command **view $locals$** or **localsview** on a C++ instance method caused the Data Explorer to display all instance variables as if they were local variables. In MULTI 7, the Data Explorer only displays actual local variables and the `this` pointer. To view the members of `this`, expand it just as you would any other structure.

Data Explorer and `print /t' Now Display Variable typedef Names

In previous versions of MULTI, displaying a variable's type via the Data Explorer's **Type** column or printing its type via the **print /t** command showed the final type name, not the `typedef` name. When used with Green Hills Compiler 2014 or later, MULTI 7 shows the `typedef` name instead of the final type name. For example, given the following C code, the Debugger displays `myNumber` (not `int`) as the type name of `intVal` and `myObj.count`:

```
typedef int myNumber;

myNumber intVal;
struct myStruct {
    char *name;
    myNumber count;
} myObj;
```

Run-Mode Partnering Now Supported with -checksum

If you are using INTEGRITY IoT or INTEGRITY version 10 or later with Green Hills Probe version 4.4 or later, making an automatic run-mode partner connection no longer sets a breakpoint in the target kernel. This allows automatic run-mode partnering even when you have built your kernel with the **-checksum** option.

This release note supersedes the INTEGRITY note that states, “The Run-mode Partner cannot be enabled in MULTI when invoking CRC or SHA-1.”

Debugger Attaches to Tasks that Hit Any-Task Hardware Breakpoint

The Debugger now automatically attaches to an unattached task if the task hits an any-task hardware breakpoint.

Register Information Now Shared

Prior to MULTI 7, each Debugger process had its own private copy of register information. As a result, changing the register configuration for a particular process would affect only that process.

Because more and more registers—especially memory-mapped registers—are being defined in modern chips, register information is now shared among some Debugger processes (register values are not shared). This reduces memory usage and speeds up the initialization of new processes in the Debugger. Register information is shared among:

- All non-TimeMachine processes in the same AddressSpace

And among:

- All processes created for homogeneous cores under a multi-core connection

Changes to register information, such as adding new registers and changing registers' bitfield structures, are reflected in all the Debugger processes that share the same register information.

Non-INTEGRITY TimeMachine No Longer Accessible from Run Mode

When used from run-mode tasks, the **TimeMachine Debugger** button ( ), **TimeMachine** menu items, and run control buttons now apply exclusively to the

INTEGRITY TimeMachine Debugger. It is no longer possible to enter non-INTEGRITY TimeMachine mode from a run-mode connection.

Child Windows Are Displayed in Front of the Debugger on Windows OS

If you are using MULTI 7 on Windows, certain Debugger child windows, such as the **Breakpoints**, **Call Stack**, and **Register View** windows, are always displayed in front of the Debugger by default. For more information, including information about modifying this behavior, see the configuration option **Debugger child windows** in “Debugger Configuration Options” in Chapter 8, “Configuration Options” in *MULTI: Managing Projects and Configuring the IDE*.

Improved `l' Command Output

The Debugger now prints better-formatted output for the **l** command. If you run any scripts that parse **l** command output, we recommend that you update them.

Issue with Windows 10 Dongle-Locked Licenses Resolved

In MULTI 7.1.6, a dongle will incorrectly report that it is not attached when used with Windows 10 updated to version 1803 (April 2018) or later. This issue is resolved with MULTI 7.1.6d. To determine if MULTI 7.1.6d is installed, you can run the **gversion** utility on **lm.exe**.

Changes to Terminology Used in the Documentation

- The operations formerly referred to in the documentation as “synchronous operations” are now called “group-based operations.”
- “Procedure(s)” and “proc(s)” are now documented as “function(s)” and “func(s).” This reflects changes made in the GUI.

Stricter Error Checking on Invalid Pointer Math Expressions

The Debugger now prints errors if you attempt to evaluate expressions that perform pointer math on addresses the Debugger cannot calculate (for example, the addresses of variables having automatic storage duration, evaluated in a context where the variable is out of scope). Such expressions previously evaluated to predictable (but usually incorrect) values.

Deprecated Features

The following features are deprecated in MULTI 7:

- `ANSICMODE` system variable
- Support for relocatable modules
- Task Manager in freeze mode
- Group-based operations, formerly referred to as “synchronous operations” (see “Changes to Terminology Used in the Documentation” on page 43)
- **prepareAllCores** configuration option
- **-allcores** option to **prepare_target** command
- **-onecore** option to **prepare_target** command
- **p** option to **l** command
- **procs** and **procedures** options to **browse** command
- **multibar** command
- **setsync** command
- **showsync** command
- **Are initially focused** setting of **Debugger child windows** configuration option (**viewsAreChildrenMode keepFocus**)
- **Save debugger window position** configuration option
- **Show version control information on file chooser dialog box** configuration option

Removed Features

The following features are not available in MULTI 7:

- **profdump** command
- **profilemode** command
- **profilereport** command

If you depend on any of the preceding features, please contact your Green Hills sales representative.

Chapter 3

Changes in MULTI 6.1.6

Contents

New Features and Enhancements 48

New Features and Enhancements

Support for Large Virtual Function Tables

MULTI 6.1.6 adds support for large virtual function tables (**--large_vtbl_offsets**)—a feature that was enabled by default in Compiler 2014.1.

Support for Freeze-Mode Debugging of INTEGRITY

MULTI 6.1.6 adds support for freeze-mode debugging of INTEGRITY 11.2.2.

Chapter 4

Changes in MULTI 6

Contents

New Features and Enhancements 50
Changes in Behavior 54
Deprecated Features 57
Removed Features 57

New Features and Enhancements

Faster Builds

Parallel builds (**Parallel Build**) are now enabled by default in the Project Manager, and there have been substantial improvements to the parallel build algorithm. Customers who were not making use of parallel builds before have observed build speedups of 4.5 times on 4-core, hyper-threaded machines. Actual speedups will vary depending on your build machine and the structure of your code base.

Ability to Retrieve More Trace Data from New SuperTrace Probes

MULTI 6 makes it easy to take advantage of the large amount of trace RAM on the SuperTrace Probe v3. As in previous versions of MULTI, you can use the **Target buffer size** option to configure the amount of trace data that the Debugger retrieves from the end of the trace buffer. However, if you later find that you need more data than was originally configured, MULTI 6 allows you to go back and retrieve it from the SuperTrace Probe v3, which always uses all available trace RAM regardless of the configured **Target buffer size**. For more information, see “Retrieving Trace Data” in Chapter 20, “Analyzing Trace Data with the TimeMachine Tool Suite” in the *MULTI: Debugging* book.

Faster Trace Processing

Trace processing performance is greatly improved in MULTI 6. In addition to general optimizations, the Debugger is now capable of making use of multiple cores to decode trace data in parallel. Parallel trace decoding is currently supported with ETMv1 and ETMv3 targets; support for other targets will be added by future Green Hills Debug Probe releases.

Trace processing is about 4 to 6 times faster than before on targets where parallel decoding is supported (with a 4-core PC), and 1.5 to 3 times faster on other targets. SuperTrace Probe v3 is required for maximum trace processing performance.

Separate Releases Allow Greater Flexibility When Upgrading

The compiler and IDE products now release independently of each other, allowing you to upgrade them separately. This makes it possible to freeze on one version of the compiler while upgrading the version of the IDE, or vice versa.

64-Bit Support in the Debugger

The standard MULTI Debugger (but not the TimeMachine Debugger) has been improved to support 64-bit targets. Specific processor support is dependent on INTEGRITY and Green Hills Probe releases.

128-Bit Hardware Registers Supported

MULTI 6 adds support for 128-bit hardware registers.

Security and Relationship Views Added to Integrate

The Integrate GUI now displays high-level relationships between AddressSpaces, allowing you to easily make sense of large, complicated INTEGRITY systems. Additionally, it helps you find security risks by showing avenues of communication between AddressSpaces. If you are using INTEGRITY IoT, see the *Integrate User's Guide* for information. If you are using an earlier version of INTEGRITY, see Appendix F, “Integrate Views” in the *MULTI: Debugging* book.

Local Variable View Updates as You Navigate the Call Stack

The Data Explorer local variable view now updates to show local variables in the new stack frame as you navigate up or down the call stack (for example by using the **E+** or **E-** command). Local variable views that you open by specifying a stack frame (for example, with **view number:$locals$**) are tied to that stack frame and still do not update when you move up or down the stack.

Ability to Restore Breakpoints

MULTI 6 gives you the ability to view and restore breakpoints that have been removed during the current MULTI debugging session. This is especially useful if you accidentally delete a breakpoint that took some time to set up (for example, if you created a breakpoint that is associated with a series of commands). For more information, see “Restoring Deleted Breakpoints” in Chapter 8, “Executing and Controlling Your Program from the Debugger” in the *MULTI: Debugging* book.

Greater Window Availability

The **Call Stack**, **Register View**, **Memory View**, and **Local Variables** windows can now be opened before your program is loaded to the target or while it is running.

New System Variables

MULTI 6 includes the new `_OS_DIR`, `_SETUP_SCRIPT`, `_SETUP_SCRIPT_DIR`, `_TOP_PROJECT`, and `_TOP_PROJECT_DIR` system variables, which can be used to allow for more complex functionality in setup scripts. For more information about these variables, see “System Variables” in Chapter 14, “Using Expressions, Variables, and Function Calls” in the *MULTI: Debugging* book.

Target List Remembers Node Expansion/Collapse

The target list now remembers node expansion/collapse for INTEGRITY applications and AddressSpaces across sessions.

Sortable Columns in the Target List

The target list can now be sorted by column. Simply click the header for the column that you want to sort by. This setting is not saved when the Debugger is closed or when new Debuggers are opened.

Improved Support for Preprocessor Macros in the Debugger

Support for evaluation of preprocessor macros in the Debugger has been rewritten to conform more closely to the C/C++ standards. The new implementation better handles expansion of nested macros and arguments containing commas, and it supports the ANSI C macro argument concatenation and stringification operators.

Browsing References Improvement

When browsing references of a C++ class's member function, you are now shown all possible call sites for the corresponding virtual functions of the class's super classes, not just the explicit references of the member function itself.

if Command Supports else if

The MULTI Debugger command **if** now supports `else if` clauses, which make it easier for you to specify a complex series of conditionally executed commands in MULTI scripts.

Licensing Improvements

A number of improvements have been made to licensing in MULTI 6. Improvements include:

- A streamlined **MULTI Licensing Wizard**, which differentiates between end user and system administrator tasks
- A new **License Information** window, which gives users and system administrators an easy way to look at usage patterns and to determine who is using licenses
- Whitelisting and blacklisting capabilities, which are useful if you want to reserve licenses for certain users, or if you want to bar particular users from obtaining licenses
- Clearer and more concise documentation

Faster Scanning of Subversion Checkouts

The MULTI 6 Checkout Browser completes scans of Subversion checkouts up to 10 times faster than before.

Changes in Behavior

IDE and compiler Installed into Separate Linking Directories

The MULTI IDE and Green Hills compiler products are now installed into separate directories that are linked as part of the installation process. If the link between the products breaks (for example, because the compiler directory is moved), you are prompted to relink them.

Upgrading to MULTI 6

If you are upgrading from an earlier version of MULTI, you must contact Green Hills Software to obtain MULTI 6 licenses.

License Clients No Longer Broadcast for a License Manager

In MULTI 5, license clients that could not obtain licenses would broadcast over the network for a License Manager from which they could obtain licenses. In MULTI 6, clients do not broadcast over the network. If a GUI program is unable to obtain a license, the **MULTI Licensing Wizard** opens so that you can request or install licenses or specify a License Manager. If a non-GUI program is unable to obtain a license, the program exits.

Changes to Computer- and Dongle-Locked Licensing

In MULTI 6, computer- and dongle-locked licenses are only available to one user at a time. If more than one user is connected at once, the license is only available from the console.

Changes to Trace Retrieval

MULTI 6 retrieves trace if you step or run backwards in the live Debugger or if you otherwise launch TimeMachine. Additionally, **Retrieve trace when target halts** is now disabled by default. These changes help prevent gaps in trace data when you halt and run with a large trace buffer.

Trace Decoding No Longer Defaults to Reading Unknown Opcodes

If a previous version of the MULTI Debugger encountered an instruction address during trace decoding that did not map to any loaded ELF files, it attempted to read the opcode from target memory by default. In MULTI 6, the Debugger aborts trace decoding by default. To revert to the old behavior, select **Read unknown opcodes from target (may halt target)** on the **Analysis** tab of the **Trace Options** window, and click **Apply**.

Changes to TS_Packet May Require TimeMachine API Application Updates

The `aid` and `tid` fields in `TS_Packet` structs of type `TSP_PID` have been changed from `uint32` to `uint64`, and the `supervisor` field has been removed. As a result, any TimeMachine API applications that inspect `TS_Packet` structs of type `TSP_PID` must be updated as follows:

- C and C++ TimeMachine API applications — Rebuild these with the updated version of ***ide_install_dir*\timemachine_api\ts_packet.h**.
- Python TimeMachine API applications — Update these to import the updated version of ***ide_install_dir*\timemachine_api\example_python\timemachine_api.py**.

Behavior of Primary and Secondary Debugger Windows

The target list is no longer available in Debugger windows that have been launched from another Debugger window. Additionally, the original Debugger window functions as a control window; when it is closed, any secondary Debugger windows that were launched from it are also closed.

Large Numbers No Longer Broken Up by Default

In MULTI 5.x and earlier, large numbers were displayed with underscores or dots to break up the numbers and ease reading. MULTI no longer breaks up large numbers by default; however, you can revert to the old behavior by using the **numberSeparator** configuration option. To use an underscore to break up large numbers in MULTI 6 and later, enter the following command in the Debugger command pane:

```
> configure numberSeparator _
```

Default Number of Scroll Back Lines Reduced

The default number of scroll back lines available in the **Cmd**, **Trg**, **I/O**, **Srl**, **Py**, and **Tfc** panes has been decreased from `524288` to `10240` to reduce memory usage. Additionally, the minimum number of scroll back lines that can be set has changed from `1024` to `10`. For more information, see the description of **cTextSize** in “Other Debugger Configuration Options” in Chapter 8, “Configuration Options” in the *MULTI: Managing Projects and Configuring the IDE* book.

Storage of PID and TID Debugger Aliases

The Debugger component previously stored 32-bit process ID (PID) and task ID (TID) aliases as signed integers and hex numbers, but now stores 64-bit PID and TID aliases as unsigned integers and hex numbers. As a consequence, previously valid PID aliases such as `debugger.pid.-1` are no longer used. Aliases are commonly specified as arguments to the **route** command. For information about the **route** command, see “Command Manipulation and Debugger Macro Commands” in Chapter 15, “Scripting Command Reference” in the *MULTI: Debugging Command Reference* book.

Deprecated Features

The following features are deprecated in MULTI 6:

- **grun** utility
- **versCtrl_RcsAutoCheckout** configuration option
- The **-filter** option to **lic_lm** and the equivalent **IP Addr Filter** field of the **License Manager Settings** dialog box (superseded by new whitelisting and blacklisting capabilities)
- Certain MULTI 5 licensing programs have been deprecated in favor of improved MULTI 6 programs. To prevent accidental use, the executables have been renamed as follows:

Deprecated Program	Discontinued Executable	Renamed Executable	Superseded by
Green Hills License Manager (GHSlm)	**lm**	**lm_compat**	**lic_lm**
glicusers utility	**glicusers**	**glicusers_compat**	**lic_userlist**
find_ghs_licenses utility	**find_ghs_licenses**	**find_ghs_licenses_compat**	**lic_find_licenses**

Removed Features

The following features are not available in MULTI 6:

- Legacy licensing technology (aka Elan)
- License Manager Web interface (superseded by a more complete license information display)
- `LMSHOW` and `LMHIDE` licensing configuration variables
- Legacy **.bld** project files
- The Project Manager menu entry **Connect → Task Manager**
- **ctags**-style tags files

- The following MULTI Editor commands and equivalent MULTI Editor menu entries:
  - **Column** command/**View → Column**
  - **Make** command/**Tools → Make**
  - **Notepad** command/**Tools → Notepad**
  - **WGUtils** command/**Tools → Launch Utility Programs**
- Hex Editor
- The **Checkout Browser** menu entry **File → Local Rescan**
- PVCS and RCS integration
- MULTI Version Control (MVC)
- Custom version control integration
- **-dotciscxx** command line option (removed from the Debugger only)
- The following MULTI Debugger commands and menu entries:
  - **C**, **Cb**, **Cu**, and **CU** commands
  - **note** command/**Tools → Notepad**
  - **Debug → Load Symbols** (Use the **loadsym** command instead. See Chapter 2, "General Debugger Command Reference" in the *MULTI: Debugging Command Reference* book.)
  - **Debug → Unload Symbols** (Use the **unloadsym** command instead.)
- **Find Function** window
- `$ent` and `$ret` special operators
- FORTRAN debugger
- Hardware Registry
- Memory filters
- RoseRT integration
- HP-UX host support
- Windows 2000 host support

If you depend on any of the preceding features, please contact your Green Hills sales representative.

Chapter 5

Changes in MULTI 5.0.6

Contents

New Features and Enhancements 60
Changes in Behavior 63
Unbundled Features 64
Deprecated Features 65
Removed Features 65

New Features and Enhancements

MULTI Project Manager

MULTI 5 introduces the MULTI Project Manager, a tool providing both the power of the traditional project builder along with new features designed to get the user up and running faster and make changes to existing projects safely and quickly.

MULTI's Project Manager gets you up and running with its new Project Wizard. Using this tool, you can create a stand-alone or INTEGRITY-based system quickly by answering straightforward questions.

Once your project has been created, the Project Manager continues to help you maintain your project as it grows and changes, allowing you to add and reconfigure components as varied as source files and INTEGRITY file systems.

To help you visually understand different aspects of your system, the Project Manager provides several different views of the project including a high level whiteboard view which can help you understand the structure of very large or complicated projects quickly.

Flexible Target Connections

Flexible target connections allow you to connect to a target once to perform a wider variety of debugging tasks. The power of a flexible connection saves you time as you explore hardware-software interaction. Using a flexible target connection, you can inspect or modify your bare target, download code or flash code to ROM, or debug code already present on the target. Using MULTI's Prepare Target Dialog, you specify what kind of access to the board you will need or how you need to modify your target's connection to the MULTI Debugger.

Improved Target Management from the Debugger

The new target list greatly improves visibility of items connected to the Debugger including bare hardware targets, multi-core systems, stand-alone programs, INTEGRITY systems, and many combinations thereof.

This list, which appears in the top pane of the Debugger window, allows you to see groupings between different tasks, address spaces, and applications for more powerful debugging of INTEGRITY systems. Each entry in the list can be selected, allowing you to debug or view that item in the source pane. This greatly reduces window clutter by allowing you to reuse a single Debugger window. Debugging multiple items side by side is as simple as double-clicking an entry in the list.

For stand-alone or bare board systems, you can now more easily connect to a board that is not yet running an application to perform board exploration or setup tasks.

Target Traffic Pane

For developers who need a behind-the-scenes look at how MULTI is interacting with their hardware, MULTI 5 introduces the Target Traffic Pane. This console, which is integrated into the tabbed pane at the bottom of the debugger window, shows all messages sent between MULTI and your hardware. This can help diagnose and/or rule out problems caused by a sensitive target's response to being probed by a debugger.

TimeMachine Enhancements

TimeMachine is more integrated into the debug environment than ever before. If your target supports TimeMachine, trace data is collected automatically in the background, allowing you to step back through code or use any other TimeMachine feature at any point in your debugging session. TimeMachine can handle many gigabytes of trace data with improved performance.

TimeMachine provides many tools which allow you to see what your system is doing. PathAnalyzer now features an array of visual aids to make understanding the path analysis information more intuitive.

You can also create your own custom TimeMachine tools using the TimeMachine API. The TimeMachine API can be accessed by your C code or Python script, allowing you to build your own analysis tools.

Improved Kernel Object Awareness

With the release of INTEGRITY 10, MULTI users will experience new levels of operating system debugging excellence. Paired with INTEGRITY 10, MULTI 5 will allow users to view INTEGRITY objects, explore relationships between INTEGRITY objects, and communicate directly with the kernel.

Global Profiling

Global Profiling provides continuously updated profiling information including CPU usage for individual address spaces and tasks. It also provides up-to-the-millisecond profiling information via MULTI's standard profiling tools.

Global Profiling is available only when debugging INTEGRITY 10 systems.

Integrated Python Engine

MULTI 5 comes equipped with a fully functional integrated python engine. Using this powerful new tool users can use MULTI in new and exciting ways.

Using a GUI building tool (TCL/TK, for example) you can build your own customized debugging windows. These windows can be viewers for special data structures, system emulators, or specialized program controllers.

Using the integrated Python engine you can control and customize the behavior of the MULTI IDE. You can specify how the MULTI IDE is launched down to window size and location.

Using the MULTI Launcher, users can execute any series of actions in the MULTI IDE. The integrated python engine expands this capability to allow the user to create programmed actions which can be executed based on dynamically available information.

Python is a complete scripting language. This flexibility and power provides all of the capabilities you might need for complete scripting including string manipulation, complete math libraries, and the power of a fully functional programming language.

Extended Version Control Support

MULTI 5 expands its version control integration to include support for the popular Subversion version control system (versions 1.3.x through 1.5.x). Additionally, support for ClearCase has been tuned to support dynamic view checkouts.

Windows Taskbar Organization

The MULTI Taskbar Organizer groups all windows from the MULTI IDE into a single entry in the Windows taskbar or tray, providing quick access to all windows. This feature is only available on Windows.

Improved Help Navigation

MULTI's new Help Viewer allows users to navigate and search the Green Hills library of documentation with greater ease, speed, and accuracy. This provides a more pleasant and powerful documentation browsing experience.

Changes in Behavior

Use of Dongle-Locked Licenses over Remote Desktop

In earlier versions of MULTI, dongle-locked licenses were unintentionally available to machines connecting via Remote Desktop to a dongle-licensed PC. In MULTI 5, dongle-locked licenses are no longer available over Remote Desktop sessions if the dongle-licensed machine allows more than one remote user connection. Dongle-locked licenses are available over Remote Desktop sessions if the remote machine allows only a single user connection.

Halt Issued Before Setup Script in Pre-5.0.6 Versions of MULTI

In versions of MULTI 5.x prior to 5.0.6, the MULTI Debugger issues a **halt** command to the target prior to running the setup script (if any). This adversely affects some hardware targets, where run control via the Debugger is not reliable after the target has been powered on, but before it has been reset. Unless you manually reset the target, the Debugger typically reports errors when trying to download a program to such a target, and the download fails.

MULTI 5.0.6 does not issue the preliminary **halt** command. If your target has a setup script, it should use the MULTI **halt** command to halt the target before reading or writing any registers or memory. This is especially important if your setup script uses the MULTI expression evaluator to write registers (for example, `$r1 = 0x12345`) instead of using debug server scripting commands (for example, `target rw r1 0x12345`).

Unbundled Features

Unbundled features are no longer included in MULTI, but are still available for separate purchase.

The following features have been unbundled from MULTI 5:

- FORTRAN debugger
- Hex Editor
- MULTI Version Control (MVC)
- Custom version control integration
- Hardware Registry Server
- Script Debugger
- PCI Devices window
- Memory filters
- OSE integration
- RoseRT integration
- Deprecated MATLAB integration interfaces. MATLAB integration is still supported through MULTI's Python interface
- Embedded Linux debugging
- HP-UX host support

Deprecated Features

The following features are deprecated in MULTI 5:

- Legacy licensing technology (aka Elan)
- PVCS, VSS, CVS, and RCS integration
- **-dotciscxx** command line option (deprecated in the Debugger only)

Removed Features

The following features are not available in MULTI 5:

- Support for using Elan dongle-locked licenses over Remote Desktop
- Function Flow tool
- The Profile window's Block Detailed report is not available with TimeMachine profiling
- **rtnserv** debug server (use multiple **rtserv** or **rtserv2** connections instead)

Chapter 6

Known Issues

Contents

Host Support 68
MULTI Installation 68
Project Manager Issues 69
Editor Issues 70
Version Control Issues 70
Target Connection Issues 72
Debugging Issues 72
Compatibility with INTEGRITY 78
MULTI-Python Integration Issues 80
File System Issues 81
Desktop Environment Issues 82

Host Support

IDE Components May Fail to Launch on Unsupported Linux Distributions

Components of the MULTI IDE may fail to launch on Linux systems using a 32-bit **libc.so** that is built assuming a 16-byte stack alignment. This problem has been observed with Manjaro Linux and may affect other distributions. If you experience it, we recommend that you switch to a supported version of Linux. For more information, see “System Requirements” on page 2.

HWE Kernels Distributed with Ubuntu May Cause Problems

Ubuntu is distributed with HWE kernels that are updated frequently and are not supported by Green Hills Software. If you encounter a problem while using an HWE kernel, we strongly advise you try a non-HWE kernel prior to contacting Green Hills Support.

Errors from Missing DLLs in Older Windows Version

In older Windows 8 installs, some IDE components may fail to launch with a dialog indicating that a run-time DLL is missing. You can fix this by downloading and installing the latest Microsoft Visual C++ Redistributable package from https://visualstudio.microsoft.com/downloads.

MULTI Installation

For known issues regarding INTEGRITY installation, see “Compatibility with INTEGRITY” on page 78.

Name of Install Directory Cannot Contain Spaces

The MULTI IDE cannot be run from a directory whose name contains spaces.

Project Manager Issues

Project Manager Performance on Very Large Project Files

Individual project files containing more than 100,000 files may be slow to load and require significant memory resources when opened in the MULTI Project Manager on Windows hosts. If insufficient resources are available, the Project Manager may exit unexpectedly.

Workaround: To improve performance, try breaking your project file into several subproject files. These files will only be loaded as needed, reducing the resources needed by the Project Manager.

Undefined __OS_DIR__ Macro When Opening Some ThreadX Projects

If you open a ThreadX project that does not include a **-os_dir** setting in the Project Manager, you may see a dialog containing the error message: “macro not defined (__OS_DIR__)”. If you see this message, adding new examples and executables to your project via the Project Manager may produce unexpected results.

Workaround: To resolve this problem, open your project, click **OK** in the error message dialog, select **Edit → Set Build Target**, and enter the path to your ThreadX installation in the **Target Selector** dialog that appears.

Projects with Build Configurations Incompatible with MULTI 6.x and Earlier

Projects that use the build configurations feature new in MULTI 7, including projects generated by the Project Wizard, cannot be opened by earlier versions of MULTI.

Truncated Field on High-DPI Displays When Creating New INTEGRITY Application Project

With some INTEGRITY versions, if the display scaling factor is higher than 100%, the **Number of Virtual Address Spaces** field in the **Configure number of Virtual AddressSpaces** dialog may be truncated and not display the full contents. This dialog appears when creating a new INTEGRITY Monolith or Dynamic Download project.

Workaround: Select the **Names of Virtual Address Spaces** option and enter a comma separated list of names for the virtual address spaces in your project.

Editor Issues

Limitation of Brace Matching with Preprocessor Directives

In the MULTI Editor, placing an opening brace inside a conditional preprocessor block may cause the closing brace to match the incorrect opening brace. This is most common when there are multiple conditional blocks and only one is expected to be compiled, for instance with `#if/#else/#endif`.

File Permissions Under Cygwin

If you use Cygwin's **chmod** to change a file's permissions to read-only, you will not be able to overwrite the file using the MULTI Editor. Cygwin does not update its version of the file system when a change is made to the Windows file system permissions.

Version Control Issues

Subversion Authentication on Windows

The MULTI Subversion integration will not properly handle password prompts on Windows when using some versions of the Subversion client if you do not already have cached authentication information. If your repository uses authentication for read or write access, and you have not yet entered a password for any Subversion operation, you are likely to encounter this problem, during which you will notice that MULTI or the Checkout Browser appears to hang.

Workaround: Perform your first password-requiring Subversion operation from the command line to ensure authentication is set up properly.

If you encounter this problem, you can recover your session by taking the following steps:

1. Kill the running **svn.exe** process. This should recover MULTI.
2. Run **svn cleanup** from the command line inside your checkout.
3. Perform the operation (probably a checkout, update, or commit) from the command line. Please consult the documentation for your specific version of the Subversion client for exact syntax.
4. Once you have successfully entered your password through Subversion's password prompt, the MULTI version control integration should work correctly.

Problems Observed with Subversion 1.4.x

Using Subversion 1.4.x is not recommended. Subversion 1.4.2 on Linux has been observed to occasionally crash while attempting to access a remote repository. Additionally, a bug in Subversion 1.4.x sometimes makes it impossible to determine whether files in a deleted directory have been committed, potentially resulting in incorrect status listings in the Checkout Browser.

Missing Version Information for libxml2.so.2

You may encounter the following error message when trying to use version control on a Linux host:

```
[path]/libxml2.so.2: no version information available
(required by ...)
```

Workaround: Remove the **libxml2.so** files that ship with MULTI and replace them with a symbolic link to the system-installed **libxml2.so.2**.

Target Connection Issues

Serial Ports Supported for Use with rtserv

On Windows, **rtserv** can only connect to serial ports with single-digit COM numbers (that is, serial ports **COM1** through **COM9**). Serial ports with two or more digits (**COM10**, for example) are not supported.

Debugging Issues

Source Pane Displays Unexpected File

If your project nests identical source trees and you attempt to navigate to or launch a particular file from the inner source tree (for example, `(1)` in the following project), the Debugger's source pane may erroneously display a file from the outer source tree (for example, `(2)` in the following project).

```
\_ nested/
| \_ src/
| | \_ file.c (1)
\_ src/
| \_ file.c (2)
```

Workaround:

1. Navigate to the **Workspaces** subdirectory of your personal configuration directory (see “Your Personal Configuration Directory” in Chapter 7, “Configuring and Customizing MULTI” in *MULTI: Managing Projects and Configuring the IDE*).
2. Delete the file that corresponds to the program you are debugging when this problem occurs. Alternatively, delete all the files in the **Workspaces** subdirectory. (Note that this does not delete any project workspaces, only saved history information that the Debugger uses when executing history commands, jumping to previous or subsequent source pane views, etc.)

Only users of a previous version of MULTI may encounter this problem.

Maximum Number of Attached Tasks

The number of tasks that the Debugger can attach to at one time is limited by the number of file descriptors that can be opened by a process in the operating system. On most Linux systems, the limit is approximately 1000. On Windows, the limit is approximately 500.

Up to 65534 Macros Supported

If a program defines more than 65534 distinct macros, MULTI may fail to evaluate expressions containing those macros.

Very Large Files Cause Slow Response Times When Single-/Double-Clicking Variables

Source files containing many lines of source code (for example, a million) may cause slow response times when you single- or double-click a variable in the Debugger source pane. This is true even when the source code is included in C/C++ source files via `#include` statements.

Workaround: Split the large file into multiple smaller files.

Limitation of Browsing Traced Memory Accesses

You can use the **tracebrowse** command in MULTI to view all reads and writes to a particular variable. However, the **tracebrowse** command only displays the memory accesses of the first four bytes of the data type. Therefore any structure, array, or class that is bigger than four bytes will only have the reads and writes to the first four bytes displayed.

Limitation of Browsing Traced Function Calls

The Trace Call Browser allows you to view all calls of a particular function and view the minimum, maximum, mean, and total duration of calls to that function. When trace data is discarded to free disk space for new data, the function calls contained in the discarded trace data are removed from the list of calls in the Trace Call Browser. The Min, Max, Mean, and Total statistical values listed in the Trace

Call Browser, however, are cumulative and take account of the discarded calls until you clear the trace data. You can clear the trace data by entering the `trace clear` command in the Debugger command pane or by selecting **TimeMachine → Clear Data** from the Debugger.

Cannot Retrieve Trace During a Blocking Run or Step

If trace retrieval is initiated while a blocking run or step operation is in progress in the Debugger, the retrieval is delayed until the blocking run or step completes. For information about blocking, see the description of the **b** parameter in "Single-Stepping Commands" in Chapter 13, "Program Execution Command Reference" in the *MULTI: Debugging Command Reference* book.

Unexpected Debug Snapshot Behavior with trace add_text_range Command

You may encounter trace decoding issues if you change trace text ranges with the **trace add_text_range** or **trace del_text_range** commands after retrieving trace data and saving a trace-only debug snapshot. You may also experience issues if you change text ranges after loading a trace-only debug snapshot and then re-decode the trace data in the debug snapshot. To avoid these issues, configure trace text ranges before retrieving any trace data that will be included in a debug snapshot. For more information, see "trace" in Chapter 19, "TimeMachine and INTEGRITY TimeMachine Command Reference" in *MULTI: Debugging Command Reference*.

Trace Configuration Files Incompatible with MULTI 4.x

The trace configuration file **trace.cfg** created by MULTI 6.0 and later is not compatible with MULTI 4.x.

Workaround: To use both MULTI 4.x and MULTI 6.0 or later on the same system:

1. Delete the **trace.cfg** file (if it exists) located in your personal configuration directory (see "Your Personal Configuration Directory" in Chapter 7, "Configuring and Customizing MULTI" in *MULTI: Managing Projects and Configuring the IDE*).
2. In the MULTI 4.x Debugger, select **Tools → Trace → Trace Options**.
3. In the Trace Options window that appears, click the **Save as Default** button.

You will only need to do this once.

Limitations of UNC Paths on Windows

Debugging projects from an INTEGRITY distribution loaded via a UNC path is not supported. Mapping the UNC path to a network drive will allow you to debug the project.

Limitations for Debugging Large Enumeration Types

The Debugger incorrectly truncates the value of an enumeration constant or of a variable of an enumeration type if the value requires more than 32 bits for representation.

Limitation of Mixing PIC or PID Code with ABS Sections

When evaluating addresses of code compiled with **-pic** or **-pid**, the Debugger first subtracts the `_TEXT` or `_DATA` offset and then determines the text or data symbol at the resulting address. If an `ABS` section happens to exist at the same address, the Debugger may incorrectly choose the symbol from the `ABS` section rather than from the PIC or PID section. This address-to-symbol mapping most frequently occurs when the Data Explorer displays addresses that point to global data such as constant strings. In this case, the Data Explorer may incorrectly name the wrong symbol as pointing to the data.

Debugging AltiVec-Specific Data Types

Power Architecture only

From the MULTI Debugger, you cannot modify the values of AltiVec-specific data types (such as vector) or cast variables to or from these types to another type. You can, however, display the values of AltiVec data types in expressions and in the Data Explorer.

Viewing Long Doubles on x86 Systems

x86 only

When viewing long doubles on native or embedded x86 systems, be advised that the Debugger rounds the value to double size before displaying it, so some precision is lost.

`update <interval>' Not Supported with INTEGRITY TimeMachine

If you are using INTEGRITY TimeMachine, the **update** command does not allow an interval to be specified. Additionally, if you use the **update** command to specify an update interval and then enter INTEGRITY TimeMachine mode, interval updates are disabled.

Limitations of History From RH850 Software Trace

RH850 only

On RH850 processors, History can be used to view function and variable information logged via the software trace features of the architecture. This feature has the following limitations:

- History is not available until trace data is retrieved from the target. MULTI will display an error saying that History is not supported until that point.
- Not supported with multi-core connections.
- Not supported in the instruction set simulator.
- Only function call information and events logged with EAGLE macros are available. Function call information is only supported when function entry and exit events are instrumented by the compiler and PC trace is disabled in the trace options.
- Timing information is only valid if timestamps are enabled in the trace options.
- Only information since the most recent trace discontinuity is shown.
- The **history -invalidate** command is not supported.
- The **eagleconfig** command is not supported.

Quitting During Command Line Function Call May Leave Program in Undefined State

Quitting the Debugger while a command line function call is in progress may leave the program you were debugging in an undefined state.

Workaround: Remove all breakpoints and allow any command line function calls in progress to complete before quitting the Debugger.

Shared Memory Problems Caused by the systemd RemoveIPC Setting

Linux only

By default, the systemd **RemoveIPC** setting deletes all IPC resources (POSIX shared memory, System V shared memory, message queues, etc.) when you log out, even if you have other programs running that are using those resources. If MULTI detects a shared memory failure that is likely linked to this setting, an error message will display in the command pane.

Workaround: Change this setting to **RemoveIPC=no** and add it to your **/etc/systemd/logind.conf** configuration file.

Daylight Saving Time May Result in Erroneous Statement

Windows only

If the MULTI Debugger is open on an executable when the time changes from standard time to daylight saving time, the Debugger may state that your executable was rebuilt and ask you to reload. This can occur even if the executable has not changed.

Symbol Names for Function Templates with Template Parameter Packs

In C++, template parameter packs (a C++11 feature) may be represented by `__parameter_pack__<...>` within the names of function templates. In such cases, parameter pack expansion is not performed within the name of the function template. Instead, the elements of the parameter pack are listed within the angle brackets that follow `__parameter_pack__`.

Compatibility with INTEGRITY

MULTI Ignores INTEGRITY 10 Installer Screen

The INTEGRITY installer's **Identify Tools Distribution** screen is used to provide the MULTI IDE with the location of the installed INTEGRITY distribution. If you are using INTEGRITY 10, the setting on this screen has no effect.

Workaround: Ignore the INTEGRITY installer's **Identify Tools Distribution** screen and the corresponding INTEGRITY documentation, and instead follow the instructions at “Configuring MULTI for Use with INTEGRITY or u-velOSity” in Chapter 2, “MULTI Tutorial” in the *MULTI: Getting Started* book.

Limitations for Loading and Unloading Applications on INTEGRITY 5.x

INTEGRITY 5.x does not support loading or unloading multiple applications at the same time, even though this can be performed through the MULTI Debugger's target list. If you attempt to load or unload multiple applications, the operation will only succeed on one of the applications.

You must wait for the previous load or unload to finish before another can be performed successfully. A load or unload has been completed successfully when the application's status changes in the target list to **Loaded** (for a load) or **Not Loaded** (for an unload).

Note that operations such as preparing your target, running, stepping, or restarting may also cause a module to become loaded. When this occurs, these operations are subject to the same restrictions.

Collecting Memory Leak Information on INTEGRITY 5.x

Collecting memory leak information when halted in the INTEGRITY kernel's `IdleTask()` is not supported and may appear to run without finishing. Pressing the **Esc** key will abort collection.

Workaround: To collect memory leak information for the kernel space `malloc` heap, you must halt your target in a C-library-enabled kernel task.

Problems with Trace of Tasks in the INTEGRITY Kernel AddressSpace

When certain probe and INTEGRITY versions are used with some BSPs, transitions between kernel tasks and the kernel itself may not be traced properly. This can cause discontinuities in kernel task call stacks and errors in the **Messages** signal in History.

RunTask May Not Have an Effect on Dynamically Downloaded Tasks

When MULTI does a dynamic download, MULTI may automatically attach to all statically defined tasks within the downloaded application as soon as they have been loaded. INTEGRITY version 5.2.1 and INTEGRITY version 5.0.7 and earlier did not allow a task to do a `RunTask` on a halted task that MULTI was attached to. This means that downloading an INTEGRITY application may result in that application being unable to start some of its own tasks.

Workaround: Run halted tasks from the Debugger.

MULTI RTSERV Auto-Connect and High Priority Tasks

If you are debugging an INTEGRITY 5.0 application with high priority tasks such that the `Idle` function will never run, MULTI will not be able to automatically start an **rtserv** connection, even if one has been requested with the Run-Mode Partner dialog or with the **set_runmode_partner** command.

Workaround: Manually establish an **rtserv** connection. It will automatically be partnered with the corresponding freeze-mode connection.

Halting or Attaching to Task Cancels a System Call with rtserv

If a task is in the middle of a host I/O system call with **rtserv** (for example, printing to the MULTI Debugger's **I/O** pane via `printf()`) and the task is halted either directly or by virtue of attaching the Debugger to the task, the system call will not complete successfully. This is most likely to occur when the task is blocked in a host I/O call, reading from standard input or from a file on the host. If you are using **rtserv2** (INTEGRITY IoT or INTEGRITY version 10 or later), halting a task does not interfere with host I/O.

System Halt with Host I/O May Make Target Unresponsive

When connected with **rtserv** to a target running INTEGRITY version 5.2.1 or INTEGRITY version 5.0.8 or earlier, and opening the OSA Explorer or otherwise invoking a System Halt, and a Host I/O operation completes after the system is halted, the target can become unresponsive to **rtserv**. This can most commonly be triggered by a target program printing to standard output immediately before the system is halted. This problem is resolved in INTEGRITY versions after 5.0.8.

Run-Mode Breakpoints May Confuse Trace

Run-mode breakpoints are implemented by replacing the targeted instruction with a special breakpoint instruction, which, when executed, causes an exception that triggers breakpoint handling. This may cause minor issues for the trace tools, which are not aware that the targeted instruction was replaced. For example, in the Trace List, the targeted instruction may be shown executing twice, and the corresponding disassembly and/or **Opcode** data may be incorrect.

MULTI-Python Integration Issues

mpythonrun Unable to Output to stdout/stderr in Cygwin

In a Cygwin environment, **mpythonrun** is unable to output data to `stdout` or `stderr`. If it attempts to do so, a dialog box appears with an error message similar to the following:

```
WriteConsole(handle=0x17 for STD_OUT_HANDLE) failed: ...
```

Clicking **OK** in the dialog box continues the execution of **mpythonrun**.

Workaround: To work around this problem, which is caused by a bug in Cygwin, redirect output to another file, or use the Windows command console or KornShell.

MULTI-Python Integration on Linux May Require Compatibility Library

To use the MULTI-Python integration with some newer Linux distributions, you will need to install a C++ compatibility library. On Debian/Ubuntu, this can be installed with the command:

```
sudo apt-get install libstdc++5
```

On Fedora Core/Red Hat Enterprise Linux, this can be installed with:

```
yum install compat-libstdc++33
```

There might be some variance in the name of the package depending on the specific version you are using. When using a version of Linux beyond those mentioned here, you will need to install a version of the GNU libstdc++v3 that is used with gcc-3.3.

File System Issues

Tilde Expansion in Cygwin

Cygwin expands the tilde symbol (~) to the value of the `HOME` environment variable as set in Cygwin. If `HOME` is set to a Cygwin path, attempts to launch non-Cygwin programs (including MULTI applications) on files whose path includes a tilde will fail.

Workaround: To work around this limitation, set the `HOME` environment variable in Cygwin to a Windows path.

Two Dots (..) Not Supported After Symbolic Links in Paths

MULTI and other Green Hills tools do not support paths that use two dots (`..`) to indicate the parent directory of a symbolic link. For example, if `symlink` is a symbolic link, the following paths are not supported:

```
/projects/symlink/..
/projects/symlink/foo/../../bar
```

NFS May Cause Poor Performance

Linux only

MULTI tools may hang if they are accessing files over an NFS server that is slow or is not responding. In certain cases, it may not be possible to terminate them, even when using `kill -9`. These problems are caused by a fundamental limitation in the design and implementation of NFS.

Workaround: To help prevent these problems, eliminate references to broken file system mount points in your user configuration directory (**~/.ghs/***) — in particular in the **integrity.dist** and/or **uvelosity.dist** files located there.

Desktop Environment Issues

Numerous Instances of MULTI Can Exhaust the Windows Desktop Heap

If you are using Windows and if many graphical processes are open simultaneously, the desktop heap can be exhausted, preventing additional graphical applications from being launched.

Workaround: If you encounter this problem, try reusing existing MULTI IDE graphical processes instead of launching new ones. For example, to reuse existing MULTI Editor processes, open new Editors by issuing the MULTI Debugger **edit** command or by selecting **File → New Editor** from the MULTI Editor menu bar.

If you continue to encounter problems, see the Microsoft Knowledge Base for information about increasing the desktop heap size.

Locking Assertion Failure May Occur Within libxcb

When running the MULTI IDE, you may encounter the following message at the console, possibly accompanied by part of the IDE exiting abnormally:

```
Locking assertion failure. Backtrace:
#0 /usr/lib/libxcb-xlib.so.0 [0xb7cfc767]
#1 /usr/lib/libxcb-xlib.so.0(xcb_xlib_lock+0x2e) [0xb7cfc81e]
...
```

This problem is caused by a known bug within the libxcb X transport binding used in some newer Linux distributions. GHS encountered this problem very infrequently during testing. If you observe it often, please contact Green Hills Support.

Taskbar Organizer and Windows 7 with Aero Peek

The Taskbar Organizer in taskbar mode is not compatible with Windows 7's Aero Peek feature: moving the mouse over the MULTI entry in the taskbar displays a white bar instead of showing window thumbnails. For information about how to disable the Taskbar Organizer, see “Configuring Taskbar Organization” in Chapter 7, “Configuring and Customizing MULTI” in the *MULTI: Managing Projects and Configuring the IDE* book.

Taskbar Organizer and nVidia Desktop Manager

The Taskbar Organizer setting to display only the Taskbar Organizer in the **Alt**+**Tab** list may not work correctly when using nVidia Desktop Management software.

Workaround: If all windows are displayed, including the Taskbar Organizer, perform the following steps:

1. Right-click the desktop and select nView Properties.
2. Click the **User Interface** tab.
3. Clear **Enable nView task switcher**.
4. Click **OK** to save your changes.

nVidia X Server with Compiz Window Manager May Crash

When using MULTI with the Compiz window manager on Linux, opening windows may cause the X server to crash. This appears to be caused by a bug which is present in version 169.12 of the nVidia driver.

To see the current version of the nVidia drivers being used, run `cat /proc/driver/nvidia/version` from the command line.

Workaround: Upgrade to version 173.14.12 or newer of the nVidia driver. For information about obtaining and installing new drivers, consult your distribution documentation or visit nVidia's website.

Alternatively, you can use a different window manager, such as Metacity.

Drawing Problems with Exceed 2008 (13.0)

You may experience problems in the MULTI Editor and Debugger command pane where text may not draw correctly when used within Exceed 2008 (version 13.0).

Workaround: Disable Exceed's **Batch Requests** option. This can be found in Exceed's configuration window under **Other Server Settings**.

Nouveau Drivers for nVidia Graphics Cards May Cause Drawing Problems

If you are using the Nouveau open-source driver for nVidia graphics cards, IDE windows may not always redraw correctly. This is due to an apparent problem in the Nouveau driver, which is used by default in certain Linux distributions, including Ubuntu 14.04 LTS.

To determine whether the Nouveau driver is in use, run the following command in a terminal:

```
grep -E -i "connected|card detect|primary dev|Setting driver" /var/log/Xorg.0.log
```

If the output contains `NOUVEAU`, the Nouveau driver is in use for the local display. If the output contains `NVIDIA`, the nVidia binary driver, which does not exhibit the problem, is in use.

Workaround: If you are using Ubuntu 14.04 LTS, follow these steps to switch to using the binary driver:

1. Launch the Dash by clicking the Ubuntu icon at the top of the Launcher.
2. Type `Additional Drivers`, and select the **Additional Drivers** entry under the **Applications** group.
3. When the **Additional Drivers** tab loads, select one of the **Using NVIDIA binary driver - version xxx.xx (proprietary)** radio buttons. If you have the option to select a **(proprietary, tested)** driver, that may be preferable.

4. Click **Apply Changes**.
5. You may be prompted to enter your login credentials for installing software on the machine.
6. After the update is complete, click **Close**.
7. Reboot the machine for the changes to take effect.

SELinux Restricts Dynamically Loaded Code

If SELinux is set to `enforcing`, it restricts dynamically loaded code to trusted modules. In enforcement mode, you may see a dialog box with contents similar to the following:

```
MULTI failed to load cpudetect.so:
cannot restore segment prot after reloc: Permission denied
```

Workaround: Temporarily disable enforcement with the following command:

```
/usr/sbin/setenforce 0
```

Alternatively, try registering the library:

```
chcon -t  texrel_shlib_t svc_cpu_ppc_so.so
chcon -t  texrel_shlib_t cpudetect.so
...
```

MULTI May Lose X Connection After Running for Several Hours

If, when running the MULTI IDE, you run a script that drives the Debugger for several hours, the IDE may exit with the following message:

```
multi: fatal: X: connection to server lost
```

This problem is caused by a known bug within libxcb and xlib in all recent Linux distributions. A fix from the library maintainers may be available in the future.