Intel® System Debugger 2017
Installation Guide and Release Notes

Installation Guide and Release Notes for Linux* host

9 May 2017

Table of Contents

1 Introduction

The Intel® System Debugger 2017 provides Linux* hosted cross-debug solutions for software developers to debug the Linux* kernel sources and dynamically loaded drivers and kernel modules on devices based on the Intel® architecture. It does so using one of the supported JTAG target probes listed under the System Requirements.

Beyond this the debugger also offers convenient and in-depth access to underlying hardware properties through a powerful graphical user interface (GUI). This makes it an ideal assistant for initial platform bring-up, firmware, BIOS, OS and device driver debugging. A set of features providing in-depth access to the development platform complete the offering for system developers:

- Execution trace support using LBR (last branch record), RTIT (real-time instruction tracing and Intel® PT (Intel® Processor Trace) for identifying incorrect execution paths or memory accesses
- Graphical representation of the page translation table with full access of the descriptor tables
- Flashing Support available upon request

These debugger capabilities minimize the time it takes to isolate and correct platform and system level problems.

This document provides system requirements, installation instructions, issues and limitations, and legal information.

Note:

Intel® System Studio is available both with and without the Intel® System Debugger. The Intel® System Studio without JTAG distributions are clearly labeled as such. To find out more about the installation and purchase options, please go to https://software.intel.com/en-us/intel-system-studio/try-buy.

1.1 Technical Support and Documentation

The installation directory of the Intel® System Studio including the Intel® System Debugger is

```
<INSTALLDIR>/system_studio_2017.x.xxx/
```

where the base directory `<INSTALLDIR>` can be:

`/opt/intel` (on a (sudo)root installation)
`$HOME/intel` (on an user installation)

The base directory can be specified by the user with a 'Custom' installation. The product however is being installed in a fixed directory structure below `<INSTALLDIR>`.

All information about the Intel® System Debugger 2017 and its components can be found under `<INSTALLDIR>/documentation_2017/en/debugger/iss2017/system_debugger/`

If you did not register your Intel® System Studio during installation, please do so at the Intel® Software Development Products Registration Center. Registration entitles you to free technical support, product updates and upgrades for the duration of the support term.

To submit issues related to this product please visit the Intel® Premier Support webpage and submit issues under the product **Intel(R) System Studio.**

Additionally you may submit questions and browse issues in the Intel® System Studio User Forum.

For information about how to find Technical Support, product documentation and samples, please visit http://software.intel.com/en-us/intel-system-studio.

Note: If your distributor provides technical support for this product, please contact them for support rather than Intel.

1.2 Product Contents

- Intel® System Debugger 2017

2 What's New in Intel® System Debugger 2017 Update 3

No Changes.

3 System Requirements

3.1 Host Software Requirements

For installation of the Intel® System Debugger, root or sudo root rights are required. With user rights, the option to install the Intel® System Debugger is not available during installation.

If you have any doubts about installation requirements, please check the Prerequisites Guide.

3.1.1 Requirements for Intel® Atom™ Processor support (start_xdb_Atom_products.sh

- 64bit (x86_64) Linux* system running
  - Red Hat Enterprise* Linux* 6, 7
  - Ubuntu* 12.04 LTS, 14.04
  - Fedora* 20, 21, 23, 24
  - SLES 12
- libusb 0.1.12 or higher
- fxload 0.0.20020411 or higher

  ```
  > sudo apt-get install fxload
  ```
  or
  ```
  > yum install fxload
  ```

- libstdc++

  To find the current library and its path, execute:

  ```
  /sbin/ldconfig -p | fgrep libstdc++
  ```

  Use this path in the next command, for example:

  ```
  > strings /usr/lib/x86_64-linux-gnu/libstdc++.so.6 | grep
  GLIBCXX_3
  ```

  If returned version list doesn't contain version greater or equal to 3.4.18 (GLIBCXX_3.4.18) please update using these commands:

  ```
  start "sudo -s" terminal
  > export http_proxy=http://<PROXY>:<PORT>
  > export https_proxy=https://<PROXY>:<PORT>
  > add-apt-repository ppa:ubuntu-toolchain-r/test
  > apt-get update
  > apt-get install libstdc++6
  ```

  If that library version is not installed on 64bit Ubuntu 12.04.5 (and other Linux versions/distros as well) you will see this error when using the System Debugger (xdb):

  ```
  Intel System Debugger 2017 (x64)
  Version 16.0.85097  (64-bit)
  Copyright(c) 2001-2015 Intel Corporation. All rights reserved.
  INFO: Initializing Target Connection Interface...
  ERROR: E-2201: Unable to connect the target communication
  interface
  ```

```
ERROR: Unable to initialize Target Connection
ERROR: Unable to initialize Target Connection Interface
```

- Java runtime environment (JRE) 1.7 or 1.6 to use the Eclipse* framework. In a web browser, access www.java.com , and download and install JRE 1.7. Make sure that the $PATH environment variable contains the path to the JRE bin-directory.

3.2 Target Software Requirements

The target platform should be based on one of the following environments:

- Bare metal, bootloader environment
- ELF DWARF2 symbol info based bootloader
- Linux*, Android*, Tizen* IVI, Wind River Linux*, Yocto Project*

3.3 Host Hardware Requirements

- Second generation Intel® Core™ i5 or i7 processor or higher.
- 2GB RAM
- 10GB free disk space for all product features and all architectures
- USB 2.0 host interface
- Intel® ITP-XDP3 JTAG Hardware Adapter

3.4.1 Target Space Requirement by Component

	Minimum RAM	Dependencies	Disk Space
xdbntf.ko	<1Mb	kernel build environment	<1Mb

3.4 Target Hardware Requirements

- Intel® In-Target Probe eXtended Debug Port
- Intel® ITP-XDP3 JTAG Hardware Adapter
- One of the following target platforms:
    - Intel® Development Kit based on the Intel® Atom™ Processors E3805, E382x, Z3680 - 2 cores (Valleyview)

- Intel® Development Kit based on the Intel® Atom™ Processors E384x, Z37xx - 4 cores (Valleyview)
- Intel® development kit based on the Intel® Atom™ Processor Z5xx (Silverthorne)
- Intel® development kit based on the Intel® Atom™ Processor x5-Z8xxx, x7-Z8700 (Cherry Trail)
- Intel® development kit based on the Intel® Atom™ Processor E6xx (Tunnel Creek)
- Intel® development kit based on the Intel® Atom™ Processor E6xx (Tunnel Creek) - overlay
- Intel® development kit based on Intel® Atom™ Processor CE42xx (Intel Media Processor CE42xx code-named Groveland)
- Intel® development kit based on Intel® Atom™ Processor CE53xx (Intel Media Processor CE53xx code-named Berryville)
- Intel® development kit based on Intel® Puma™ 6 Media Gateway CE26xx (Cat Mountain)

- TinCanTools* FLYSWATTER2 and Olimex* ARM-USB-OCD-H support the following platform
  - Intel® Quark™ SoC X1000

- **Note:** Please find more details about debugger startup support for specific JTAG probe and target CPU/Platform combinations under section *5.1 Starting the Debugger.*

3.5 Ordering JTAG Device for Intel® System Debugger

3.5.1 Intel® ITP-XDP3

To order the Intel® ITP-XDP3 device, please

1. Go to https://designintools.intel.com/, select the `Debug Tools` product category and add `ITP-XDP BR3` to your cart.
2. or contact the Hibbert Group* at Intelvtg@hibbertgroup.com and request the VTG order form.

We will also gladly assist with the ordering process.If you have any questions please submit an issue in the Intel® System Studio product of Intel® Premier Support https://premier.intel.com or send an email to IntelSystemStudio@intel.com.

3.5.2 TinCanTools* FLYSWATTER2

Go to http://www.tincantools.com/wiki/Compiling_OpenOCD

3.5.3 Olimex* ARM-USB-OCD-H

Go to https://www.olimex.com/Products/ARM/JTAG/ARM-USB-OCD-H/

The following pin adapter was used to connect the Intel® System Debugger to the Intel® Quark™ SoC board: https://www.olimex.com/Products/ARM/JTAG/ARM-JTAG-20-10/

4 Installation

The default installation directory of the Intel® System Studio 2017 and its component Intel® System Debugger 2017 are

```
<INSTALLDIR>/system_debugger_2017/
```

Please check the host software requirements closely before proceeding with the installation.

If you have any doubts about installation requirements, please check the Prerequisites Guide.

4.1 Product Installation (Online Installer)

The Intel® System Debugger 2017 on Linux* host component of the Intel® System Studio 2017 is available as part of a downloadable online installer. If you only intend to install the Intel® System Debugger you can thus reduce the package size that is downloaded for the actual install. Using the online installer requires to be connected to the internet and that https protocol based component downloads are permitted by your firewall.

Execute the downloaded online install script

```
> system_studio_2017.0.xxx_online.sh
```

following all instructions.

4.2 Product Installation (Full Product)

The Intel® System Debugger 2017 on Linux* is part of the Intel® System Studio 2017 downloadable installer. For installation of the debugger on the development host please follow the steps below:

1. Unpack the downloaded tool suite package in a directory to which you have write access.
   ```
   > tar -zxvf system_studio_2017.0.xxx.tgz
   ```
2. Change into the directory the tar file was extracted to
   ```
   cd ./ system_studio_2017.0.xxx/
   ```
3. Execute one of the installation scripts for command line installation or using the GUI installer.
   ```
   >./install.sh
   >./install_GUI.sh
   ```
4. To be able to install the Intel® System Debugger it is necessary to select "install as root" or "install as root using sudo". Without root privileges the option to install the Intel® System Debugger will not be offered during install.
5. To start the Intel® System Debugger change into the

   `<INSTALLDIR>/system_debugger_2017/system_debugger` installation directory.

   From there run the debugger launch shell script that best fits your host-target setup.

4.3 Silent Install

For information on automated or “silent” install capability, please see http://intel.ly/ngVHY8.

Please note that the Intel® ITP-XDP3 JTAG device driver installation does not support silent install. If you choose silent installation for the Intel® System Debugger, this device driver will need to be installed separately afterwards.

4.4 Installing Intel® XDP3 JTAG Probe

If the install_GUI.sh installation script is executed using root access, su or sudo rights, the required drivers will be installed automatically. Root, su or sudo rights are required for the installation.

4.5 Uninstalling the Debugger

To uninstall, simply go to the Intel® System Studio installation directory `<INSTALLDIR>/system_studio_2017.0.xxx/` and run the `uninstall.sh or uninstall_GUI.sh` script you find there.

5 Intel® System Debugger Usage Notes

5.1 Starting the Debugger

To start the Intel® System Debugger change into the `<INSTALLDIR>/system_debugger_2017/system_debugger/` installation directory.

From there run the debugger launch shell script that best fits your host-target setup.

1. `start_xdb_Atom_products.sh` - this script allows the user to debug Intel Atom family processors using the Intel® ITP-XDP3.

    - Intel Puma 6 Media Gateway CE26xx (Cat Mountain)
    - Intel Media Processor CE42xx (Groveland)
    - Intel Media Processor CE53xx (Berryville)
    - Intel Atom Processor E6xx (Tunnel Creek)
    - Intel Atom Processor Z5xx (Silverthorne)
    - Intel Atom Processor E3805, E382x, Z3680 - 2 cores (Valleyview)
    - Intel Atom Processor E384x, Z37xx - 4 cores (Valleyview)
    - Intel Atom Processor x5-Z8xxx, x7-Z8700 (Cherry Trail)

2. `start_xdb_gdb_remote.sh` – this script launches the debugger for connection to the

    - Intel Quark SoC

    using the OpenOCD* interface.

The debugger is now running and will establish a debug connection to the powerd-on target device.

5.2 Handling of Target Access Issues

Please ensure that the target system is running a recent firmware version allowing JTAG debugging. If a “no threads found” error message appears, and applying the restart recipe (see below) doesn’t help the situation, please contact customer support for that specific platform.

If target access issues occur, please apply the following recipe for a complete reset of all the components that take part in JTAG debugging, in this order:

1. Close the debugger.
2. Unplug **both** the USB and power connectors from the XDP3 probe **at the same time**, so no cable is connected to the XDP3 probe anymore.

3. Using the task manager, kill the process MasterFrame.HostApplication.exe (on Linux-hosted systems, there's a mono process running it, so kill that process).
4. Power off the target system.
5. Plug the USB and power connectors back into the XDP3 probe.
6. Power on the target system.
7. Start the debugger.

5.3 Troubleshooting Target Stability issues

5.3. 1 Considerations

The Intel® System Debugger requires a great deal of data to construct a full source-level view of the target state. For a simple operation such as step, there can be tens to hundreds of individual accesses to target state. In the case where one of these accesses crashes the target, it can be difficult to identify the exact root cause.

5.3.2 Techniques for isolating causes

1. Close as many GUI panels as possible

Each panel in the debugger (e.g., the registers panel, MSR panel, etc.) is self-updating. If many panels are open, then there are many target accesses. Closing panels is one way to isolate which functional group is causing the problem.

2. Use the TCI_LOG flag to turn on API logging

The debugger can log all transactions made to the communication backend. Although this information is primarily intended for developers, it may also be useful for identifying root causes of stability issues. Enable logging by setting the environment variable TCI_LOG=1.

For example, you can edit the `start_xdb_Atom_products.sh` startup script to include the command

```
set TCI_LOG=1.
```

On request from you Intel support team you may additionally include the command

`set ENGINE_LOG=1`, which will however slow down debugger startup considerably to provide very detailed logs.

This technique causes a log file to be created in the folder `/home/<name>/.sysdbg_2017`, where <name> is the user login.

This technique causes a log file to be created in the startup folder. By default this folder is `/opt/intel/system_studio_2017.0.xxx/debugger/system_debugger`

5.4 OS Awareness / Kernel Module Debugging

The Linux* OS awareness pulldown menu allows visibility of all currently active kernel threads. It also provides the ability to view a list of all currently loaded kernel modules with status information and memory location of initialization methods and cleanup methods. Setting whether to stop the target and commence debugging a kernel module on module load, initialization or cleanup/exit allows to start debugging a kernel module and loading its symbolic information. You can then set your breakpoints at the function entry points of the kernel module you want to debug, release the target using the *run* command and trigger an event that will cause the breakpoint to be hit to start your actual debug session.

You do not need to select kernel modules that are already loaded, but can add additional kernel module names to the list of kernel modules that are monitored and have the debugger stop at load, initialization or cleanup just as it would with the kernel modules that are already populated in the OS awareness pulldown menu as they were loaded during the Linux* OS boot process.

To debug kernel modules the following steps additional to selecting or adding a kernel module in the module list are necessary.

In a debugger script or in the debugger console window enter the following commands:

SET DIRECTORY "<kernel module path>"

This path setting is necessary to enable the automatic source and symbol info mapping upon kernel module load as described above.

To use this feature for runtime loaded kernel module debugging you will need to have the kernel module **xdbntf.ko** running and installed on the target device. The folder `<install-dir>/system_studio_2017.0.xxx/debugger/system_debugger/kernel-modules/xdbntf` contains code to generate a Linux* kernel module that enables kernel module debugging with the Intel® System Debugger.

For generation simply transfer these files to your target system and invoke make. This will generate the kernel object **xdbntf.ko**.

To enable module debugging this object has to be loaded prior to starting the debugger via the command **insmod xdbntf.ko**. After finishing the debug session, the module can be unloaded with **rmmod xdbntf**.

5.5 Troubleshooting Simics Simulated Target

5.5.1 Connection fails with error message: "E-2201 TCP/IP Socket initialization or connection failed: Connection to host localhost:9123 failed, please verify server is running, and that the address and port are correct"

The error indicates that the Intel® System Debugger could not connect to the Simics software. This can be caused by:

1. There is a local firewall running which permits TCP accesses to localhost:9123
2. The license for Simics is expired. Either update to the latest version of Intel® System Debugger or contact technical support.

6 Issues and Limitations

6.1 Supported targets on Red Hat Enterprise* Linux* is limited

The following target connections are not working on Red Hat Enterprise* Linux:

- Intel® DCI USB 3.x Debug Class (USB3 cable)
- Simics (Demo Mode)

6.2 Debugger start silently fails on Fedora* 21 with default OpenJDK under Gnome/Unity/KDE

After starting a debugger script, a splash screen is only displayed, but the debugger exits silently. No diagnostic is issued.

To overcome this issue you may apply one of the following workarounds:

6.2.1 Install and use JRE from Oracle

Download jre-8u25-linux-x64.tar.gz from www.oracle.com
http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html
Unzip the package and add the /jre1.8.0_25/bin to PATH

6.2.2 Change GTK theme to “Raleigh”

On KDE use lxappearance
Install, goto “SystemSettings->Application Appearance->GTK+Appearance” and change widget style to “Raleigh”

On Unity/Gnome use gtk-theme-switch2
Install and launch the gui, switch to "Raleigh"

Both utilities can be installed from the packet manager.

6.3 Debugger start fails on Fedora* 23 and Fedora*24

Using the debugger on Fedora* 23 and 24 requires a downgrade of java and to run it using sudo. To downgrade Java to 1.7 follow these steps:

1. Download http://download.oracle.com/otn-pub/java/jdk/8u60-b27/jre-8u60-linux-x64.rpm
2. Install with: sudo dfn install jre-8u60-linux-x64.rpm
3. Add to alternatives: sudo alternatives --install /usr/bin/java java /usr/java/jre1.7.0_79/bin/java 1
4. Choose 2nd option from alternatives: sudo update-alternatives --config java
5. This should let you select the newly installed 1.7 java as option 2
   Verify that it switched by doing:
   java -version
   This should display 1.7 instead of 1.8
6. Run the debugger using sudo:
   sudo ./xdb.sh

6.4 Target-specific issues:

6.4.1 Functional differences of 60-pin vs. 10-pin JTAG

Some platforms (e.g. the Intel® Quark™ SoC X1000 based Galileo board) do not implement the full 60-pin debug port that is traditionally used on Intel systems, in this case the functionality of the debugger will be limited, especially in the following areas:

- Detection of reset by the debugger
- Initiation of reset by the debugger
- Halting the target at the reset vector

6.4.2 Platform reset policy may inhibit debugger operation

Some platforms implement reset in such a way that the debugger may not be able to gain control of the target immediately after reset. This impacts the debugger operation in the following ways:

- The debugger may not be able to restore breakpoints after reset; the breakpoints may appear “enabled” in the GUI but will not in fact be enabled in the target.
- The debugger will not halt automatically at the reset vector; if the user wishes to debug early in the boot process there will be no way to manually initiate a halt quickly enough.

6.4.3 Platform security policy may inhibit debugger operation

In some platforms the security policy may disable JTAG access to the CPU, this is intended to prevent reverse-engineering. In this case the Intel® System Debugger will successfully connect to the target, however it will not be able to discover any CPUs on the JTAG bus, and will warn the user that no CPUs are available. To resolve this issue please ensure that that platform firmware has enabled access to the CPUs via JTAG, this is typically done by flashing a special “debug” firmware into the target.

Also note that in some cases the CPU or CPU module may have physically disabled JTAG access, especially in production or near-production versions. In this case please work with the platform business unit to obtain JTAG-enabled hardware.

6.4.4 Target power management and platform power policy on tablet systems may inhibit debugger operation

On some tablet designs the platform architecture includes aggressive power-management of the CPU, this will impact debugger operation in that a CPU in a low-power state cannot be accessed via JTAG. This will manifest in a variety of ways:

- Error messages indicating that “no threads are available”
- Error messages indicating that “target could not halt”
- No threads displayed in the hardware threads window

In general these problems will be mitigated by doing one or more of the following:

- Ensure that the CPUs are in an active state when the debugger is first started (e.g. in early boot firmware where no power management is present)
- Ensure that the OS has a workload that will inhibit low power states (e.g. play a video, run animated wallpaper, etc.)

- Disable low-power states in the platform when possible (generally a BIOS setting)

6.4.5 Platform reset implementation may limit debugger-initiated reset:

Debugger-initiated reset is not an industry standard feature, it is implemented using sideband signals on the 60-pin Intel XDP port, and it is subject to the reset implementation on the target system. Some targets may not reset reliably via the debugger's reset/restart feature, this will typically result in a message such as "`WARNING: target did not halt after reset, forcing a halt`" being displayed in the debugger console, followed by additional error messages. In this case the user may need to manually initiate a reset on the target via buttons, debug card, etc.

6.4.6 Platform reset implementation may limit ability to halt at reset vector:

Halting the CPU at the reset vector (first instruction fetched) is a CPU/platform dependent feature and may be limited due to target implementation details. The main impact to the user is:

- **inability to debug early platform boot code due to a runaway target.** In this case it may be necessary to build a special firmware with a hard-coded infinite loop early in the boot flow.
- **inability of the debugger to re-apply breakpoints after target reset.** In this case the debugger may show breakpoints as "enabled" in the GUI, but they will not be installed in the target, the user should manually halt the target, disable, re-enable breakpoints to ensure they are applied correctly.

6.5 Intel® Atom™ Processor Z3xxx and E3xxx specific issues

6.5.1 Platform power management policy may limit debugger control of the target:

The platform power-management policy may include power management of the CPUs, this may limit availability of debugger features when the threads are in a low-power state. Examples include:

6.5.2 Launching the debugger when the target is off or in a low-power state may cause unexpected behavior

The debugger needs to see all threads when connecting to the target in order to correctly initialize both the debugger state, and the debug resources in the target. The debugger will report the number and type of threads observed during initialization (e.g. "`INFO: Connected to Processor type: <name> (4 threads)`") the user should confirm that this matches the expected configuration. Resetting the target (with the debugger running)

should clear the issue and cause all expected threads to show up in the hardware threads window.

6.5.3 Hardware Threads window may show no threads/partial threads/disabled threads

Hardware Threads window may show no threads/partial threads/disabled threads due to the CPUs appearing/disappearing from the JTAG scanchain, and/or showing up in the JTAG scanchain as “disabled”. This condition should only happen when the target is in “run” state, when halted (e.g. from hitting a breakpoint) the debugger should correctly show all threads. ***if the debugger halts and all threads are not shown*** then the debugger is in an incoherent state and you may need to restart your debug session.

6.5.4 User-initiated Halt may occasionally return errors, especially if the target is in a low-power state.

Errors include “`E-2201: Target has no active threads, this operation is not permitted.`” and “`E-2201: Target did not halt execution.`”

This condition occurs when the debugger attempts to halt the target, but the CPUs are asleep and therefore unresponsive to debugger commands. Workaround: manually bring the CPUs out of sleep (e.g. by fiddling with the tablet) ***or*** try repeatedly to halt via the debugger, after 2-3 tries the target typically wakes up.

6.5.5 Kernel module load configurations may be unreliable

Due to limitations in silicon debug features, kernel module load notifications may not function correctly on Intel® Atom™ Processor Z3xxx and E3xxx based platforms. The following should be observed to work around this:

- **Software breakpoint in the target must be set prior to using xdbntf.** The breakpoint should be in an unused/unreachable code location, it is not necessary that the breakpoint is ever hit, its purpose is to enable the Intel® System Debugger redirection logic in the Silicon, which will allow the kernel module notifications to function correctly.
- **in some cases the notification will only partially work**: this will manifest as a hung system, with the Intel® System Debugger indicating a “running” state. In this case the user should manually halt the target, at which point the debugger will detect the notification, consume it, and resume target execution.

6.6 General Feature Limitations

6.6.1 Opening Help content on system with Java* JRE 1.7.x can lead to Java SEGV

Opening "Help > Help Contents" can lead to a Java segmentation fault triggered by the Intel® System Debugger user interface. Workaround is to access the Intel® System Debugger User's Guide at https://software.intel.com/en-us/xdb_2017_ug instead.

6.6.2 Debugger cannot be launched in user mode once it has been launched as root

The Intel® System Debugger root installation requirement implies that if the debugger is launched as root instead of regular user only once, access permissions will not permit launch as regular user afterwards.

Message: "daltempinfo.txt" owned by root:root => "Cannot update registration of MasterFrame: Access denied for writing to '/usr/share/Intel/DAL/MasterFrame'"

Workaround: Install as root and exclusively run debugger as normal user (or exclusively run debugger as root, but do not switch root/normal user)

Workaround: (after 1st invocation of XDB as root) run this command in root shell: "chmod -R ugo+rw /usr/share/Intel/"

6.6.3 PCI bus scan in PCI tool very slow on Linux* host

PCI bus scan in the PCI tool add-on is very slow on Linux* host and interrupting the scan can lead to debugger hang. It is recommended to only scan small bus ranges (e.g. 0-10) and to not Help contents loading from within debugger not supported on Ubuntu* 14.04 with Java* JRE 1.7.x

6.6.4 Help contents loading from within debugger is not supported on Ubuntu* 14.04 with Java* JRE 1.7.x.

Invoke help from terminal shell with the following command:

```
$ firefox <INYSTALLDIR/
documentation_2017/en/debugger/iss2017/system_debugger/cl/index.ht
m
```

6.6.5 Only installation as "root" or "sudo" user supportd

Currently, only the "root" and "sudo" installation types are supported using the Linux* hosted Intel® System Studio installer. The "user" installation type should not be selected on Linux hosted systems.

6.6.6 Support for Intel® Atom™ Processor bitfield editor register views

To receive information on how to access bitfield editor chipset register views for Intel® Atom™ Processor s, please send an email to IntelSystemStudio@intel.com for details.

6.6.7 Locals Window updates can be slow

The local window updates may be slow in many cases where it is evaluating many large structs or in scopes with many locals. If the slowness is noticed, it is recommended to close the locals window.

6.6.8 Kernel Threads Window Population Slow

The Linux* OS awareness plug-in for the Intel® System Debugger includes a Kernel Threads Window, that displays all current kernel threads and information about their state. When the Kernel Threads Window is opened it can take several seconds before the actual content is displayed. The initial window content of “No data.” will disappear once kernel thread data is available. This can take up to 20 seconds.

6.6.9 Debugger puts a file system lock on symbol files

Currently when a symbol file is loaded in the debugger a file system lock is placed on this file to prevent other processes from deleting or modifying this file. If this lock is preventing you from recompiling your program, simply use the Unload feature found in the Load Dialog. Unloading a symbol file will release the file system lock and allow you to modify or delete the symbol file without exiting the debugger.

6.6.10 Function and file information not listed for watchpoints in breakpoint window

When setting a data breakpoint (watchpoint) the breakpoint listing in the breakpoint window does not contain file and function information for the data breakpoint.

6.6.11 Local variables and evaluation windows do not display multi-dimensional arrays correctly

Multi-dimensional arrays are displayed as vectors in the debugger's local variables window and evaluations window suppressing one dimension of the array.

6.6.12 Evaluation window for global variables may be missing type information

When evaluating a global variable in a debugger evaluation window the variable name and it’s value are displayed, but type information may not be displayed.

6.6.13 Writing to a non-writable vector registers may incorrectly update register value display

In the vector register window it is possible that a write to a vector register seems to have been successful, when a new value was entered from within said window, despite the register being non-writable at the time and not actually having been updated.

6.6.14 Root or Sudo Access required for Intel® System Debugger Install

To be able to install the Intel® System Debugger it is necessary to either launch the tool suite installation script install_GUI.sh with root privileges or to select "install as root" or "install as root using sudo" during the Intel® System Studio installation process. Installation without root privileges will not be successful.

6.6.15 Memory Writes to Uninitialized Memory

Memory writes to uninitialized or read-only memory (this includes setting software breakpoints or accessing memory mapped registers) can lead to a crash of the target or a loss of the target control. The debugger will not prevent these memory accesses when requested by the user (e.g. changing instructions in the disassembly window).

6.6.16 Flash Writer disables pre-existing Breakpoints

Flashing the BIOS will disable all code breakpoints and data breakpoints you may have had set prior to using the flash writer.

6.6.17 Master Flash Header Read/Write not supported for Intel® Atom™ Processor CE4200

On the Intel® Atom™ Processor CE4200 the Master Flash Header serves as a road map for the contents of flash that are processed by security and host firmware. It contains the location and size of each element in the flash, as well as a list of host firmware images that the security processor will attempt to boot. Currently the flash writer plug-in for the Intel® System Debugger does not support writing or modifying the Master Flash Header.

It is of course possible to use the terminal Master Flash Header commands mfhlist mfhinfo and mfhinit in conjunction with the Intel® System Debugger flash writer plug-in.

mfhlist provides the location of the Master Flash Header entries and where the current platform boot configuration expects the various flash images to be put. It’s output can be used as a guidance for setting the start address when using the flash writer plug-in.

For NOR Non-Trusted Boot and NAND/eMMC Non-Trusted boot can be configured such that target boot is possible even if no Master Flash Header is present on the platform.

eMMC Trusted Boot does require the presence of a Master Flash Header and requires that the actual memory layout does match its contents.

Please read the Platform User Guide closely for further details on the Master Flash

Header and its usage.

6.6.18 ERROR: E-2201: Invalid thread/device number selected

Running the Intel® System Debugger on **64-bit Fedora** Linux requires using a correct java version. Download and install Oracle Java jdk-7u75-linux-x64 (or jre-8u40-linux-x64.rpm) and point the path to those versions in start_XXXX.sh directly:

- /usr/java/jdk1.7.0_75/bin/java -jar ...

or

- /usr/java/jre1.8.0_40/bin/java -jar ...

6.6.19 Opening the web help ("Help"=>"Help Contents") on a Linux host OS might cause the debugger to crash.

This can be worked around by uninstalling the package libwebkitgtk-1.0-0 using the Linux distribution's package manager. Alternatively, the documentation can be opened from the "Intel® System Debugger Documentation Index" page.

7 Change History

7.1 Intel® System Debugger 2017 Update 2

7.1.1 New Features

- Added the platform breakpoint types: Shutdown Break and Machine Check Break
- For code lines that are mapping to multiple assembly blocks the context menu provides now also the options "Go Here", "Set Current Location" and "Jump to assembler".
- Improved the messaging for cases when no additional hardware breakpoints are available.
- Updated Simics
- The context menu in the assembler and source window allows now to set a software or hardware breakpoint explicitly.
- The Script file: `<installdir>scripts/startup.xdb` was added to allow execution of commands automatically during startup of Intel® System Debugger

- Variables are now marked as `<optimized out>` if not available because of compiler optimizations and as `<currently unavailable>` if they are out of scope.
- UEFI Buttons are loaded automatically. No need to manually execute efi.xdb anymore
- Improved DWARF Version 4 debug information support.

7.1.2 Bug Fixes

- Corrected masking of base addresses in paging structures.
- Added correct handling of Memory Protection Key IDs in paging structures.
- For the BATCH command the option /NOOUT was fixed.
- Complex debug information decoding improved especially for Memory Reference Code (MRC) debugging.

Setting breakpoints on source lines that map to multiple assembly locations is working now also if the source file is not part of the first symbol file loaded.

7.2 Intel® System Debugger 2017 Update 1

No changes

7.3 Intel® System Debugger 2017 Initial Release

7.3.1 EFI Script:

New script "EFI.xdb" adds UEFI-specific helper buttons to the user interface to discover PEI/DXE phase debug symbols.” In order to use the script, use either of these methods:

- Via the menu of the debugger GUI: Click File -> Execute Command File and point it to the file “efi.xdb” which is contained in the debugger’s “scripts” subdirectory.
- Alternatively, on the debugger command line, use this command: BATCH “scripts/efi.xdb”

This will add three new buttons to the debugger’s toolbar:

- LoadThis: This will parse memory around the current instruction pointer and load symbolic source information for the currently executing UEFI module.

- LoadPEIMs: This will search the flash firmware volume and load debug information for all the PEI-phase modules.
- LoadDXEModules: This will search DRAM for the system table, and load debug information for all the DXE-phase modules that have been loaded by the target up to this point.

Please note that any of these commands may sometimes lead to the debugger's disassembly window showing "No data". This can be fixed by right clicking into the window and selecting "Reload" from the context menu.

7.3.2 System Management Mode (SMM) debug support

Improved the support for SMM debugging (currently only available for Intel® DAL based target connections). See the "Debugging System Management Mode" chapter in the user documentation for details and usage instructions.

- Special stepping handling for RSM (Return from System Management Mode) instruction: Stepping over the RSM instruction will now use SMM Exit Break (and not have the target freely running).
- SMM Entry Break and SMM Exit Break available to set from the GUI: Use the "Platform" tab in the "Create Breakpoint" dialog. Also these special breaks can be set by commands SET BREAKPOINT ON "SMM Entry Break" or SET BREAKPOINT ON "SMM Exit Break", respectively

7.3.3 Single Startup

Single startup, "xdb.bat" and "xdb.sh" are added. This provides a single interface to connect to different targets and platforms. The previous startup scripts are still present in case of issues with the new startup script.

8 Attributions

This product includes software developed at:

The Apache Software Foundation (http://www.apache.org/).

Portions of this software were originally based on the following:
- software copyright (c) 1999, IBM Corporation., http://www.ibm.com.
- software copyright (c) 1999, Sun Microsystems., http://www.sun.com.
- the W3C consortium (http://www.w3c.org) ,
- the SAX project (http://www.saxproject.org)
- voluntary contributions made by Paul Eng on behalf of the
Apache Software Foundation that were originally developed at iClick, Inc.,

software copyright (c) 1999.

This product includes updcrc macro,
Satchell Evaluations and Chuck Forsberg.
Copyright (C) 1986 Stephen Satchell.

This product includes software developed by the MX4J project
(http://mx4j.sourceforge.net).

This product includes ICU 1.8.1 and later.
Copyright (c) 1995-2006 International Business Machines Corporation and others.

Portions copyright (c) 1997-2007 Cypress Semiconductor Corporation.
All rights reserved.

This product includes XORP.
Copyright (c) 2001-2004 International Computer Science Institute

This product includes software from the book
"Linux Device Drivers" by Alessandro Rubini and Jonathan Corbet,
published by O'Reilly & Associates.

This product includes hashtab.c.
Bob Jenkins, 1996.

9 Disclaimer and Legal Information

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at Intel.com, or from the OEM or retailer.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel, the Intel logo, Xeon, and Xeon Phi are trademarks of Intel Corporation in the U.S. and/or other countries.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice Revision #20110804

*Other names and brands may be claimed as the property of others