Intel® System Studio
Installation Guide and Release Notes
Intel® JTAG Debugger 3.0

Installation Guide and Release Notes

Document number: 322074-013US

13 February 2013

Contents

1 Introduction 3
Technical Support and Documentation 4
Ordering required JTAG Device 5
1.1.1 Intel® ITP-XDP3 5
1.1.2 Macraigor* usb2Demon* 5
Product Contents 5
2 What's New 6
Target Platform Support 6
3 System Requirements 7
Host Software Requirements 7
Target Software Requirements 7
Host Hardware Requirements 7
Target Hardware Requirements 8
Host-Target Platform Support Matrix 9
4 Installation Notes 10
Installing Intel® XDP3 JTAG Probe 12
Installing Macraigor Systems* usb2Demon* Support 12
Uninstalling the Debugger 13
5 Features 14
5.1.1 Full Intel LPIA Silicon & Chipset support 14
5.1.2 Execution Trace Support 14
5.1.3 Updated Source Files Window 14

5.1.4 OS Awareness / Kernel Module Debugging 14
5.1.7 Unload of Symbol Information 15
5.1.8 Power Events Handling 15
5.1.9 eMMC NAND Flash Programming on the Intel® Atom™ Processor CE42xx and CE53xx 15
5.1.10 NAND Flashing support on Intel® Puma6™ Media Gateway 16
5.1.11 Supported Flash Types 16
5.1.6 Scripting Language 16
5.1.7 Page Translation Table 16
6 Usage Notes 17
6.1.1 Starting the Debugger 17
6.1.2 Do not use soft links for Intel® JTAG Debugger installation directory 17
6.1.3 Burning blank flash on Intel® Atom™ Processor CE5300 based platforms 17
6.1.4 Enabling Run-Time Loaded Kernel Module Debugging 18
6.1.5 OS Awareness / Kernel Module Debugging 18
6.1.6 Unload of Symbol Information 19
6.1.7 eMMC NAND Flash Programming on Intel® Atom™ Processor CE42xx and CE53xx 19
6.1.8 eMMC Flash Recovery on Intel® Puma6™ Media Gateway 19
7 Issues and Limitations 21
Known Issues and Limitations 21
7.1.1 No offline license activation for Intel® JTAG Debugger without license file. 21
7.1.2 Support for Intel® Atom™ processor bitfield editor register views 21
7.1.3 Macraigor Systems* usb2Demon* does not support single stepping on Intel® Puma6™ Media Gateway 21
7.1.4 Locals Window updates can be slow 21
7.1.5 Kernel Threads Window Population Slow 21
7.1.6 Debugger puts a Windows file system lock on symbol files 21
7.1.7 Frequently loading and unloading debug symbols 21
7.1.8 Flash programming with empty flash parts (CE4200) 22
7.1.9 Use of Macraigor* usb2Demon to debug Intel® Atom™ processor CE4xxx based platforms may require board changes 22
7.1.10 Older Macraigor* usb2Demon 60pin connector headers 22
7.1.11 Backup of large flash partitions may fail 22

7.1.12 Verification of flash content on Intel® Atom™ processor E6xx based platform.....22
7.1.13 Function and file information not listed for watchpoints in breakpoint window 22
7.1.14 Target doesn't boot when Macraigor* usb2Demon* JTAG device is not initialized 23
7.1.15 Local variables and evaluation windows do not display multi-dimensional arrays correctly 23
7.1.16 Evaluation window for global variables may be missing type information 23
7.1.17 Writing to a non-writable vector registers may incorrectly update register value display 23
7.1.18 With Intel® Hyper-Threading Technology disabled in BIOS debugger SMP configuration must be disabled as well when using Macraigor Systems* usb2Demon* 23
7.1.19 Root or Sudo Access required for JTAG Debugger Install.....23
7.1.20 Memory Writes to Un-Initialized Memory.....24
7.1.21 Flash Writer disables pre-existing Breakpoints.....24
7.1.22 Master Flash Header Read/Write not supported for Intel® Atom™ Processor CE4200 24
8 Attributions.....25
9 Disclaimer and Legal Information.....26

1 Introduction

The Intel® JTAG Debugger 3.0 provides Linux* hosted cross-debug solutions for software developers to debug the Linux* kernel sources and dynamically loaded drivers and kernel modules on Intel® Atom™ Processor based devices. It does so using the In-Target Probe eXtended Debug Port (ITP-XDP) on Intel® Atom™ Processor (N2xxx, D2xxx, E6xx, CE42xx, and CE 53xx).

As JTAG communication device you have the choice between the Macraigor usb2Demon* and the Intel® ITP-XDP3. 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 bringup and firmware and BIOS debugging. A set of features providing in-depth access to the development platform complete the offering for system developers:

- Execution trace support for identifying incorrect execution paths or memory accesses
- Bitfield editors for comprehensive and bit level access to silicon platform features

- Graphical representation of the page translation table with full access of the descriptor tables
- Flash writer plug-in that allows to update the BIOS or firmware as well as flash-resident OS images

These debugger capabilities minimizes 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.

Technical Support and Documentation

The directory `<install-dir>/system_studio_2013.0.xxx/documentation/debugger` includes these release and installation notes `jtag-rlease-notes.pdf`.
The QuickStart Guide `xdb_quickstart_lin.pdf` and debugger Usage Guide `xdb_usage.pdf` can be found at `<install-dir>/system_studio_2013.0.xxx/documentation/en_US/debugger/xdb/first_use`.

In addition, the Intel® JTAG Debugger Processor includes online help provided in form of the document titled "Intel® Debugger Online Help", which is available from the Help menu of the graphical user interface (GUI) as well as `<install-dir>/system_studio_2013.0.xxx/documentation/en_US/debugger/xdb/first_use/debugger_documentation.htm`.

If you did not register your compiler 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 for Linux* OS**.

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.

Ordering required JTAG Device

1.1.1 Intel® ITP-XDP3

To order the Intel® ITP-XDP3 device, please contact the Hibbert Group* at Intelvtg@hibbertgroup.com and request the VTG order form.

1.1.2 Macraigor* usb2Demon*

Go to http://www.macraigor.com/usbDemon.htm and select the Intel(R) Atom™ Processor target with the appropriate 24, 31 or 60 pin connector for your target device.

Product Contents

- Intel® JTAG Debugger 3.0, Build [79.xxx.x]

2 What's New

Below are some of the new features in the Intel® JTAG Debugger 2.3 for Intel® Atom™ Processor

Target Platform Support

- **Support for Yocto Project* 1.2.x and 1.3 based Linux* OS Intel® Atom™ Processor N2xxx and D2xxx**

  The Intel® JTAG Debugger for Intel® Atom™ Processor has been tested against debugging the Yocto Project* 1.2.x and 1.3 operating system software stack running on Intel® Atom™ Processor N2xxx and D2xxx based platforms.

- **Support for Yocto Project* 1.2.x and 1.3 based Linux* OS on Intel® Atom™ Processor E6xx**

  The Intel® JTAG Debugger for Intel® Atom™ Processor has been tested against debugging the Yocto Project* 1.2 and 1.2.1 operating system software stack running on Intel® Atom™ Processor E6xx based platforms.

- **Support for Intel® Atom™ Processor CE53xx**

  The Intel® JTAG Debugger has been tested against debugging CE Linux running on Intel® Atom™ Processor CE53xx based platforms.

- **Support for Intel® Puma6™ Media Gateway**

  The Intel® JTAG Debugger has been tested against debugging CE Linux running on Intel® Puma6™ Media Gateway

- **Support for GUI assisted Flash Writer Plug-in for Intel® Atom™ processor CE53xx**

  The Intel® JTAG Debugger provides flashing support through its Flash Memory Tool plugin providing easy graphical user interface assisted access to reprogramming the NOR as well as NAND flashes on software development platforms based on the Intel®

- **Support for Intel® Atom™ processor E6xx system register bitfield editor view**

  The Intel(R) JTAG Debugger provides access to chipset registers on the application processor as well as the Intel I/O Hub. All public registers of the Intel(R) Atom(TM) processor E6xx are accessible through the bitfield editor views offered by the Intel(R) JTAG Debugger.

3 System Requirements

Host Software Requirements

- For installation of the Intel® JTAG Debugger, root or sudo root rights are required. With user rights, the option to install the Intel® JTAG Debugger is not available during installation.
- Linux* system running
    - Red Hat Enterprise* Linux* 5, 6
    - Ubuntu* 10.04 LTS, 11.04, 11.10, 12.04 LTS
    - Fedora* 14, 15, 16, 17, 18
    - openSUSE 12.1
    - SUSE LINUX Enterprise Server* 10 SP4, 11 SP2
- libusb 0.1.12 or higher
- fxload 0.0.20020411 or higher
- Linux32 OCDRemote v9.7-2 or newer is required for Macraigor Systems* usb2Demon* support.
  We recommend using OCDRemote* 9.7-2 for Intel® Atom™_Processor CE53xx and version 9.9-x of the Macraigor* Systems* driver for all other intended target platforms. Support for the latest Intel® Atom™ processor based platforms may require newer driver versions. (check for recommended version at http://www.macraigor.com/Intel/)
- Java runtime environment (JRE) 1.5 or 1.6 to use the Eclipse* framework. In a web browser, access www.java.com , and download and install JRE 1.6. Make sure that the $PATH environment variable contains the path to the JRE bin-directory.
- Kernel development sources for your development host system will be required for Macraigor System* driver installation

Target Software Requirements

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

- Yocto Project * 1.1, 1.2, 1.2.1, 1.3 based Linux* OS for Intel® Atom™ Processor Z5xx, E6xx, N2xxx, D2xxx
- Wind River* Linux* 5 based Linux* OS for Intel® Atom™ Processor E6xx, N2xxx, D2xxx
- CE Linux* PR28 or newer for Intel® Atom™ Processor CE4xxx, CE53xx or Intel® Puma6™ Media Gateway
- Bare metal, BIOS, firmware environment for all supported hardware targets

Host Hardware Requirements

- IA-32 or Intel®64 architecture based host computer supporting Intel® Streaming SIMD Extensions 3 (Intel® SSE3) instructions (1st generation Intel® Core™ Processor Family), or compatible non-Intel processor.
    - For the best experience, a multi-core or multi-processor system is recommended.
- 1GB RAM (2GB recommended)
- 4GB free disk space for all product features and all architectures

- USB 2.0 host interface

Target Hardware Requirements

- In-Target Probe eXtended Debug Port
- One of the following target platforms:
    - Intel® development kit based on the Intel® Atom™ processor N2xxx or Intel® Atom™ processor D2xxx
    - Intel® development kit based on the Intel® Atom™ processor Z5xx
    - Intel® development kit based on the Intel® Atom™ processor E6xx
    - Intel® development kit based on Intel® Atom™ processor CE42xx
    - Intel® development kit based on Intel® Atom™ processor CE53xx
    - Intel® development kit based on Intel® Puma6™ Media Gateway
- Macraigor Systems* usb2Demon* (http://www.macraigor.com/usbDemon.htm) JTAG hardware adapter or alternatively Intel's ITP-XDP3 JTAG Hardware Adapter

Host-Target Platform Support Matrix

Table 1. Intel® JTAG Debugger

Host / Target	Z5xx	E6xx	CE42xx	CE53xx	N2xxx/ D2xxx	Puma6
Fedora* 17	McG, XDP	McG, XDP	McG, XDP	McG, XDP	McG, XDP	McG, XDP
Ubuntu* 11.04	McG, XDP	McG, XDP	McG, XDP	McG, XDP	McG, XDP	McG, XDP

McG: Macraigor Systems* usb2Demon

XDP: Intel(R) ITP-XDP3

[+]: Macraigor Systems* OCDRemote driver installation may require workaround on latest kernel versions. Please consult Known Issues and Limitations section.

4 Installation Notes

The default installation directory is
`/opt/intel/system_studio_2013.0.xxx/debugger/xdb`

For installation of the debugger update on the development host please follow the steps below:

1. Unpack the tool suite package in a directory to which you have write access.
   `> tar -zxvf l_cembd_p_2013.0.xxx.tgz`
2. Upon registering for the program you will receive a serial number and email with a license file. You will need either of these two to complete the installation process. If you want to use the license file you can point to it during install, but you can also copy it to `/opt/intel/licenses/`for automatic pickup by the installer.
3. Change into the directory the tar file was extracted to `../l_cembd_p_2013.0.xxx`
4. Execute the install script in the directory where the tar file was extracted.
   `>./install.sh`
5. If you are not logged in as root, you will be asked if you want to install as root, install as root using sudo, or install without root privileges. Installing as root (using sudo if you have that privilege) is recommended, as that will update the system RPM database. Use the `install as current user` option if you want to install to a private area. To be able to install the Intel® JTAG Debugger however it is necessary to select "install as root" or "install as root using sudo". Without root privileges the option to install the Intel® JTAG Debugger will not be offered during install. Depending on the installation location of the Yocto Project* Application Development Toolkit the integration of the Intel® C++ Compiler with it may require root privileges as well.
6. The welcome message to the Intel® System Studio for Linux* appears along with an outline of the installation process. Press the `Enter` key to continue.
7. The installation routine checks for the availability of all product dependencies. Please take care of these dependencies, if a warning message appears.
8. Afterwards you will be asked to read the end-user license agreement for the tool suite. Press the `Enter` key to continue with reading the license agreement. Once done type `accept` to continue with the installation.
9. When asked whether you would like to activate and install your product select one of the options provided depending on whether you have a license file available or not. If there is already a valid license file available and installed on your system, the installation routine will recommend to simply use the existing license file. If you do not have access to the internet at the time of installation, select the alternative activation option. Please note thatoOffline activation of the Intel® JTAG Debugger without license file is currently not supported. For the Intel® JTAG Debugger you will need to download the license file from the registration center https://registrationcenter.intel.com prior to offline installation and use activation via license file to achieve installation without being connected to the internet.

10. The next screen will ask you whether you would like to participate in the Intel® software Improvement Program. This will help us identify opportunities for product improvement. If you do not want to participate simply select option [2].
11. The following screen lets you review your installation options. If you would like to only install a specific tool suite component, select [3] and change the components settings. Else, continue with the default choice [1] to start the installation.

```
Step no: 5 of 7 | Options
--------------------------------------------------------------------------------
You are now ready to begin installation. You can use all default installation
settings by simply choosing the "Start installation Now" option or you can
customize these settings by selecting any of the change options given below
first. You can view a summary of the settings by selecting
"Show pre-install summary".
--------------------------------------------------------------------------------
1. Start installation Now
2. Change install directory      [ /opt/intel ]
3. Change components to install  [ All ]
4. Show pre-install summary
h. Help
b. Back to the previous menu
q. Quit
--------------------------------------------------------------------------------
Please type a selection or press "Enter" to accept default choice [1]:
```

12. The installation will be finalized and you will be informed when the installation has been completed successfully.

Below is a list of ready to go target connection configurations:

```
start_xdb_MCRG_Z500.sh    Intel® Atom™ Processor Z5xx   Macraigor* usb2Demon*
start_xdb_XDP3_Z500.sh    Intel® Atom™ Processor Z5xx   Intel® ITP-XDP
start_xdb_MCRG_E600.sh    Intel® Atom™ Processor E6xx   Macraigor* usb2Demon*
start_xdb_XDP3_E600.sh    Intel® Atom™ Processor E6xx   Intel® ITP-XDP
start_xdb_MCRG_N2000.sh   Intel® Atom™ Processor N2xxx  Macraigor* usb2Demon
                          Intel® Atom™ Processor D2xxx
start_xdb_XDP3_N2000.sh   Intel® Atom™ Processor N2xxx  Intel® ITP-XDP
                          Intel® Atom™ Processor D2xxx
start_xdb_MCRG_CE4100.sh  Intel® Atom™ Processor CE41xx Macraigor* usb2Demon*
start_xdb_XDP3_CE4100.sh  Intel® Atom™ Processor CE41xx Intel® ITP-XDP
start_xdb_MCRG_CE4200.sh  Intel® Atom™ Processor CE42xx Macraigor* usb2Demon*
start_xdb_XDP3_CE4200.sh  Intel® Atom™ Processor CE42xx Intel® ITP-XDP
start_xdb_MCRG_CE5300.sh  Intel® Atom™ Processor CE53xx Macraigor* usb2Demon*
start_xdb_XDP3_CE5300.sh  Intel® Atom™ Processor CE53xx Intel® ITP-XDP
start_xdb_MCRG_CE2600.sh  Intel® Puma6™ Media Gateway   Macraigor* usb2Demon*
start_xdb_XDP3_CE2600.sh  Intel® Puma6™ Media Gateway   Intel® ITP-XDP
```

To have the installer define a default xdb.sh configuration start script that points to one of these, please consult the advanced installation options for the Intel® JTAG Debugger.

Installing Intel® XDP3 JTAG Probe

If the install.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.

Installing Macraigor Systems* usb2Demon* Support

To enable support for the Macraigor Systems* usb2Demon* device for debugging Intel® Atom™ processor based platforms with the Intel® JTAG Debugger it is necessary to install the Linux* drivers for the Macraigor Systems* usb2Demon* device. The driver can be found at http://www.macraigor.com/full_gnu.htm. Please scroll down all the way and download and install the Linux32 OCDRemote package recommended at http://www.macraigor.com/intel/ . Follow the installation instructions posted at the same location closely.

1. For Fedora* host installations download mcgr-hwsupport-x.x-x.x86_64.rpm or mcgr-hwsupport-x.x-x.i386.rpm from http://www.macraigor.com/full_gnu.htm. For Ubuntu* host installations you can download mcgr-hwsupport-x.x-x.x86_64.deb or mcgr-hwsupport-x.x-x.i386.deb from the same location. You will find the latest recommended version of the OCDRemote* driver set referenced there.

   The Intel® JTAG Debugger has been validated for use with the Macraigor Systems* usb2Demon* device and OCDRemote* 9.90. We recommend using at least version 9.90 of the Macraigor* Systems* driver for most platforms except for the Intel® Atom™ Processor CE53xx, where we currently recommend v9.72.

Ensure you have root or sudo access rights on your Linux* installation.

2. Install mcgr-hwsupport-x.x-x.x86_64.rpm or mcgr-hwsupport-x.x-x.i386.rpm using the `rpm -i` command,

   or install mcgr-hwsupport-x.x-x.x86_64.deb or mcgr-hwsupport-x.x-x.i386.deb using the `dpkg –install` command .

3. You should now be able to launch the Intel(R) JTAG Debugger using the respective start_xdb_MCRG_processorname.sh startup script. These scripts are being put into the /opt/intel/system_studio_2013.0.xxx directory by the tool suite installer as described in the general installation section above.

For further details on how to configure the OCDRemote* driver set from Macraigor* Systems, please refer to the full installation instructions at http://www.macraigor.com/eclipse/eclipse_faq.htm.

Uninstalling the Debugger

To uninstall, simply go to the bin directory of the debugger component you would like to uninstall `/opt/intel/system_studio_2013.0.xxx/debugger/xdb/bin` and run the corresponding `uninstall.sh` script you find there.

5 Features

5.1.1 Full Intel LPIA Silicon & Chipset support

Provides an in-depth view into Intel® Atom™ Processor chipsets. Supports silicon specific features, including architectural registers, Intel® Streaming SIMD Extensions 3 (SSE3), as well as Graphics Chipset Register support through JTAG. Graphical representation of peripheral registers and bit fields with online documentation are available for select Intel® Atom™ Processor based platforms.

5.1.2 Execution Trace Support

Enables viewing of execution history and thus enhances understanding of the flow of an executed program allowing the analysis of the execution path to find errors and identify the root cause for exceptions.

5.1.3 Updated Source Files Window

The Source Files Window now supports tree views. It shows all source files compiled into the loaded debuggee. The source files are displayed in a tree structure. At top level the libraries, below that the folders and as leave nodes the source files itself are displayed. There is a search text field at the top of the window, which filters the window content. A text filter can be any expression including wildcards. Double click on a source file in this tree will open the source file in a Source Window.

5.1.4 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 command:

SET DIRECTORY "<kernel module path>"

or add the kernel module directory using the `Options>Source Directories` debugger menu entry. 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 **/opt/intel/system_studio_2013.0.xxx/debugger/xdb/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**.

5.1.7 Unload of Symbol Information

To unload a symbol file, open the Load dialog. Click the “Unload Symbol File” tab and select the symbol file to be unloaded. This will remove the symbol information from the debug session. Removing symbol information is useful in order to load different or new symbol information.

5.1.8 Power Events Handling

The debugger can properly handle externally controlled power events without needing to close the debugger. If a target is reset or powered-off the debugger will identify the “Target power loss.” Once power is restored the debugger will attempt to halt the target at the reset vector.

Power event handling on Intel® Atom™ Processor N2xxx/D2xxx with Macraigor Systems* usb2Demon* may be limited.

5.1.9 eMMC NAND Flash Programming on the Intel® Atom™ Processor CE42xx and CE53xx

5.1.9.1 Partitions

There are 3 partitions defined on eMMC. Our flash tool is capable of programming each of the 3 partitions: boot1, boot2 and the user partition.

5.1.9.2 Addressing

Addressing space for each partition is independent. Therefore you should address the beginning of each partition as address 0. Currently 32KB is the minimum write or read size that we support.

5.1.9.3 Erasing

Block erasing is not implemented for this flash. eMMC flash can be written to without erasing first. Therefore the only erase we support is on the entire partition.

Note- Currently “Verification” is not supported on eMMC flash. Please do a backup and manual binary diff the two files.

5.1.10 NAND Flashing support on Intel® Puma6™ Media Gateway

The Intel® JTAG Debugger supports flashing eMMC NAND partitions on Intel® Atom™ Processor CE4xxx, CE5xxx and Intel® Puma6™ Media Gateway based reference platforms.

5.1.11 Supported Flash Types

Below is a brief description of the supported flash types for the specific target platforms:

Intel® Atom™ procesor E6xx

- SPI-FLASH

Intel® Atom™ procesor N2xx/D2xxx

- SPI-FLASH

Intel® Atom™ Processor CE4xxx,CE53xx

- NOR Flash
- eMMC NAND Flash
- eMMC Boot Partitions
- eMMC User Partition

Intel® Puma6™ Media Gateway:

- SPI-Flash
- eMMC NAND Flash
- eMMC Boot Partitions
- eMMC User Partition

5.1.6 Scripting Language

Create a batch file based on a rich set of powerful debugging script language commands and execute it in non-interactive mode. Results can be logged and analyzed afterwards.

5.1.7 Page Translation Table

Instant and simple resolution and translation between physical and virtual address space.

6 Usage Notes

6.1.1 Starting the Debugger

To start the Intel® JTAG Debugger for Intel® Atom™ Processor change into the

<*install-dir*>/system_studio_2013.0.xxx/debugger/xdb/ directory

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

Below is a list of ready to go target connection configurations:

```
start_xdb_MCRG_Z500.sh    Intel® Atom™ Processor Z5xx   Macraigor* usb2Demon*
start_xdb_XDP3_Z500.sh    Intel® Atom™ Processor Z5xx   Intel® ITP-XDP
start_xdb_MCRG_E600.sh    Intel® Atom™ Processor E6xx   Macraigor* usb2Demon*
start_xdb_XDP3_E600.sh    Intel® Atom™ Processor E6xx   Intel® ITP-XDP
start_xdb_MCRG_N2000.sh   Intel® Atom™ Processor N2xxx  Macraigor* usb2Demon
                          Intel® Atom™ Processor D2xxx
start_xdb_XDP3_N2000.sh   Intel® Atom™ Processor N2xxx  Intel® ITP-XDP
                          Intel® Atom™ Processor D2xxx
start_xdb_MCRG_CE4100.sh  Intel® Atom™ Processor CE41xx Macraigor* usb2Demon*
start_xdb_XDP3_CE4100.sh  Intel® Atom™ Processor CE41xx Intel® ITP-XDP
start_xdb_MCRG_CE4200.sh  Intel® Atom™ Processor CE42xx Macraigor* usb2Demon*
start_xdb_XDP3_CE4200.sh  Intel® Atom™ Processor CE42xx Intel® ITP-XDP
start_xdb_MCRG_CE5300.sh  Intel® Atom™ Processor CE53xx Macraigor* usb2Demon*
start_xdb_XDP3_CE5300.sh  Intel® Atom™ Processor CE53xx Intel® ITP-XDP
start_xdb_MCRG_CE2600.sh  Intel® Puma6™ Media Gateway   Macraigor* usb2Demon*
start_xdb_XDP3_CE2600.sh  Intel® Puma6™ Media Gateway   Intel® ITP-XDP
```

To have the installer define a default xdb.sh configuration start script that points to one of these, please consult the advanced installation options.

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

6.1.2 Do not use soft links for Intel® JTAG Debugger installation directory

This path into which the Intel® JTAG Debugger engineering release is unpacked is automatically detected. However this may fail with soft-links, so please avoid using these when extracting the contents of the tar ball.

6.1.3 Burning blank flash on Intel® Atom™ Processor CE5300 based platforms

Burning blank flash, requires that the SPI boot path is enabled and that the board is not configured to boot from eMMC.
If the board is configured to boot from eMMC and the eMMC flash is empty than the target is held in reset. This is indicated by an active yellow LED on the board. There are switches on the board that are used to select the boot path, on the platform code-named “Mt. Carmel” Fab B, these are switches SW3F1 switch 6 and switch 7. For SPI boot, switch 6 should be OFF and switch 7 should be on. Please ensure that SPI boot is enabled if the a blank flash is to be written.

6.1.4 Enabling Run-Time Loaded Kernel Module Debugging

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 **/opt/intel/system_studio_2013.0.xxx/debugger/xdb/kernel-modules/xdbntf** contains code to generate a Linux kernel module that enables kernel module debugging with the Intel system debugger. The same code is also included in the system_studio_target.tgz target install package of the Intel® System Studio .

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**.

6.1.5 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 command:

SET DIRECTORY "<kernel module path>"

or add the kernel module directory using the `Options>Source Directories` debugger menu entry. 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 **/opt/intel/system_studio_2013.0.xxx/debugger/xdb/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**.

6.1.6 Unload of Symbol Information

To unload a symbol file, open the Load dialog. Click the “Unload Symbol File” tab and select the symbol file to be unloaded. This will remove the symbol information from the debug session. Removing symbol information is useful in order to load different or new symbol information.

6.1.7 eMMC NAND Flash Programming on Intel® Atom™ Processor CE42xx and CE53xx

6.1.7.1 Partitions

There are 3 partitions defined on eMMC. Our flash tool is capable of programming each of the 3 partitions: boot1, boot2 and the user partition.

6.1.7.2 Addressing

Addressing space for each partition is independent. Therefore you should address the beginning of each partition as address 0. Currently 32KB is the minimum write or read size that we support.

6.1.7.3 Erasing

Block erasing is not implemented for this flash. eMMC flash can be written to without erasing first. Therefore the only erase we support is on the entire partition.

Note- Currently “Verification” is not supported on eMMC flash. Please do a backup and manual binary diff the two files.

6.1.8 eMMC Flash Recovery on Intel® Puma6™ Media Gateway

For flashing a blank eMMC flash on a platform based on the Intel® Puma6™ Media Gateway we provide a special flash recovery script in conjunction with the Intel® JTAG Debugger GUI Flash Memory Tool plugin.

Please follow the steps below to recover eMMC flash or write to blank eMMC flash memory on a platform based on the Intel® Puma6™ Media Gateway:

1. Power cycle the board

Launch `start_xdb_MCRG_CE2600_eMMC_Flash_Recovery.sht` (Macraigor* usb2Demon*) or `start_xdb_XDP3_CE2600_eMMC_Flash_Recovery.sh` (Intel® ITP-XDP3) in a terminal window from the `/opt/intel/system_studio_2013.0.xxx/debugger/xdb` directory.

2. Wait for the "`Flash Recovery completed successfully!`" message on the command line and then exit.
3. If using the Macraigor* usb2demon, it is now recommended to power the target off and back on again.
   Launch `start_xdb_MCRG_CE2600.sh` in
4. `/opt/intel/system_studio_2013.0.xxx/debugger/xdb` from a command line window Close assembler window
5. Open flash plugin
   a. Select CE2600 in Board dropdown and eMMC-user in the Flash dropdown box
6. Set Data file to your CEFDK binary (ce2600.bin), set Offset to 0x80800 and select burn
7. Restart target and it should boot to CEFDK.

7 Issues and Limitations

Known Issues and Limitations

7.1.1 No offline license activation for Intel® JTAG Debugger without license file.

Offline activation of the Intel® JTAG Debugger without license file is currently not supported. For the Intel® JTAG Debugger you will need to download the license file from the registration center https://registrationcenter.intel.com prior to offline installation and use activation via license file to achieve installation without being connected to the internet.

7.1.2 Support for Intel® Atom™ processor bitfield editor register views

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

7.1.3 Macraigor Systems* usb2Demon* does not support single stepping on Intel® Puma6™ Media Gateway

The Macraigor Systems* usb2Demon* driver OCDRemote* 9.9-0 does currently not properly support halt on breakpoint and single step on the Intel® Pume6™ Media Gateway. It is recommended to use the Intel® ITP-XDP3 device until an updated OCDRemote* distribution is available.

7.1.4 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.

7.1.5 Kernel Threads Window Population Slow

The Linux* OS awareness plug-in for the Intel® JTAG 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.

7.1.6 Debugger puts a Windows 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.

7.1.7 Frequently loading and unloading debug symbols

Reloading symbol information numerous times can cause an “Out of Memory” exception and crash the debugger. Due to memory leaks, reloading symbol information numerous times

without exiting the debugger can cause the debugger to use more than the maximum allotted amount of memory. This will cause a fatal exception and will crash the debugger. A work around for this issue is to exit the debugger if the memory usage becomes high.

7.1.8 Flash programming with empty flash parts (CE4200)

In order for the flash burning algorithm to execute the target must first be restarted and sitting at the reset vector. This is normally handled by the flash plugin. However, on CE4200 boards that do not have programmed flash parts, the platform's reset does not function. Therefore, when programming these boards, a manual reset is required.

To successfully burn flash on platforms with blank flash, a user should follow these steps:

1) Connect the debugger, and verify that the target is halted
2) Manually reset the target by pressing the reset button on the target
3) Verify that the EIP is not pointing at 0xFFF0
4) Open the flash plugin dialog and resume normal steps to program the flash.

7.1.9 Use of Macraigor* usb2Demon to debug Intel® Atom™ processor CE4xxx based platforms may require board changes

On some of the Intel reference platforms for Intel® Atom™ processor CE4100 the RefDes serial resistor R6D20 may need to be replaced with a 0 Ohm resistor. On some of the Intel reference platforms for Intel(R) Atom(TM) processor CE4200 the RefDes serial resistor R4E5 may need to be replaced with a 0 Ohm resistor. Please refer to the board schematics and the platform design guides for details.

7.1.10 Older Macraigor* usb2Demon 60pin connector headers

The reference voltage and signal voltage pins on Macraigor* usb2Demon 60pin connector headers from prior to September 2010 are connected. Since those two voltages differ from each other on the Intel® Atom™ Processor E600 reference platform it is strongly recommended to only use recently purchased Macraigor* usb2Demon 60pin connector headers with these platforms. Not following this recommendation can damage the reference platform.

7.1.11 Backup of large flash partitions may fail

It is recommended to avoid using the flash writer plug-in to backup large flash partitions (>200Mb) from the Intel® Atom™ Processor onto the workstation.

7.1.12 Verification of flash content on Intel® Atom™ processor E6xx based platform

Verifying flash content may only be possible immediately after writing it. When the system boots, it might modify the flash content (e.g. saving changed settings). When the flash content will be compared to the original file that got written to the flash, the verification will show differences.

7.1.13 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.

7.1.14 Target doesn't boot when Macraigor* usb2Demon* JTAG device is not initialized

If the Macraigor* probe is connected to the host system the first time, it doesn't get initialized until a debugger is using the probe. Without initialization the probe, the JTAG pins are in an undefined state and can prevent the target system from booting. The might lead to unstable behavior on the target including failing boot sequence. After the debugger has initialized the probe the target system should boot without problems.
This does not affect the XDP3 connection. XDP3 probe is automatically initialized when it gets plugged into the host system.

7.1.15 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.

7.1.16 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.

7.1.17 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.

7.1.18 With Intel® Hyper-Threading Technology disabled in BIOS debugger SMP configuration must be disabled as well when using Macraigor Systems* usb2Demon*

When Intel® Hyper-Threading Technology is disabled in the BIOS, the debugger SMP configuration must be turned off as well. The Macraigor System* driver interface returns an error condition on disabled processor cores that is indistinguishable from other error conditions for the debugger. If SMP support is disabled, the debugger will not try to access the second hyper-thread and thus the error condition will not occur.

To disable SMP support please edit *xdb.sh* in the */opt/intel/system_studio_2013.0.xxx/debugger/xdb* directory and change *-tgttype 'JTAG IA SMP'* to *-tgttype 'JTAG IA'* in the java launch command line.

With current Macraigor Systems* drivers the above mentioned error condition may rarely also occur if on an SMP enabled system one hyper-thread is halted during debugger memory read/write operations.

7.1.19 Root or Sudo Access required for JTAG Debugger Install

To be able to install the Intel(R) JTAG Debugger it is necessary to either launch the tool suite installation script install.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.

7.1.20 Memory Writes to Un-Initialized Memory

Memory writes to un-initialized 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).

7.1.21 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.

7.1.22 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® JTAG 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® JTAG 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.

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 licensed from Macraigor Systems, LLC.
Copyright (c) 2004-2009, Macraigor Systems LLC. All rights reserved.

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

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.
UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.
Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.
The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.
Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.
Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. Go to: http://www.intel.com/products/processor_number/

MPEG-1, MPEG-2, MPEG-4, H.261, H.263, H.264, MP3, DV, VC-1, MJPEG, AC3, AAC, G.711, G.722, G.722.1, G.722.2, AMRWB, Extended AMRWB (AMRWB+), G.167, G.168, G.169, G.723.1, G.726, G.728, G.729, G.729.1, GSM AMR, GSM FR are international standards promoted by ISO, IEC, ITU, ETSI, 3GPP and other organizations. Implementations of these standards, or the standard enabled platforms may require licenses from various entities, including Intel Corporation.

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino Inside, Cilk, Core Inside, i960, Intel, the Intel logo, Intel AppUp, Intel Atom, Intel Atom Inside, Intel Core, Intel Inside, Intel Inside logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel Sponsors of Tomorrow., the Intel Sponsors of Tomorrow. logo, Intel StrataFlash, Intel Viiv, Intel vPro, Intel XScale, InTru, the InTru logo, InTru soundmark, Itanium, Itanium Inside, MCS, MMX, Moblin, Pentium, Pentium Inside, skoool, the skoool logo, Sound Mark, The Journey Inside, vPro Inside, VTune, Xeon, and Xeon Inside are trademarks of Intel Corporation in the U.S. and other countries.

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

Microsoft, Windows, Visual Studio, Visual C++, and the Windows logo are trademarks, or registered trademarks of Microsoft Corporation in the United States and/or other countries.

Java is a registered trademark of Oracle and/or its affiliates.