Intel® System Studio
Embedded Use Case Notes

Use Case Release Notes

Document number: 322069-014US

10 February 2013

Contents

1 Introduction

This document provides a brief overview over the different ways the Intel® System Studio 2013 for Linux* supports embedded development and cross-development. Embedded platforms come in many shapes and sizes. Many are based on heterogeneous System-on-Chip designs using a multitude of hardware architectures. System software stack customization and application development for varied configurations requires a flexible approach and development tools that can be adapted to the needs of cross-development for a variety of software stacks.

The Intel® System Studio consists of multiple components for developing, debugging, tuning and deploying system and application code targeted towards embedded designs.

The tool suite covers several different use cases targeting development for embedded intelligent system platforms ranging from Intel® Atom™ Processor based low-power embedded platforms to 2nd and 3rd generation Intel® Core™ microarchitecture based designs for high throughput data processing and signal processing. Accordingly you may want to install different components of the tool suite in different ways.

use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

2 Online Resources

Software Migration

1. Software Migration Design Guide

   http://www.intel.com/p/en_US/embedded/designcenter/migration/powerpc/software-migration?iid=1928

Embedded

1. Intel® Embedded
   http://www.intel.com/p/en_US/embedded?firstlogin=true

   Embedded Community
   http://embedded.communities.intel.com/community/en

2. Intel® Embedded Development Tools
   http://software.intel.com/en-us/embedded-development-tools

3 Build Environment

Eclipse* IDE Integration

3.1.1 Installation

The Intel® C++ Compiler can be automatically integrated into a preexisting Eclipse* CDT installation. The Eclipse* CDK, Eclipse* JRE and the Eclipse* CDT integrated development environment are not shipped with this package of the Intel® System Studio. The Intel® System Studio contains an interactive script that asks for the Eclipse component location and helps to integrate already installed Intel® System Studio components into the Eclipse* IDE. Currently only integration into the 64bit (x86_64) build of Eclipse* is supported.

To have the Eclipse Integration done as part of the Intel® System Studio installation navigate to

```
/opt/intel/system_studio_2013.0.xxx/eclipse_support/
```

and execute the automated Eclipse integration script.

```
> ./eclipse_integration.sh
```

If asked point the integration script to the installation directory of your Eclipse* 3.7 install. Usually this would be `/opt/eclipse/`.

The prerequisites for successful Eclipse integration are:

1. Eclipse* 3.7 (Indigo)
2. Eclipse* CDT 8.0
3. Java Runtime Environment (JRE) version 6.0 (also called 1.6) update 11 or later.

The Eclipse* 3.7.x environment that the integration has been validated against and that is strongly recommended to be used can be downloaded from the eclipse.org website at

http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/indigo/SR2/eclipse-cpp-indigo-SR2-incubation-linux-gtk-x86_64.tar.gz (64bit host)

3.1.2 Launching Eclipse for Development with the Intel C++ Compiler

Since Eclipse requires a JRE to execute, you must ensure that an appropriate JRE is available to Eclipse prior to its invocation. You can set the `PATH` environment variable to the full path of the folder of the `java` file from the JRE installed on your system or reference the full path of the java executable from the JRE installed on your system in the `-vm` parameter of the Eclipse command, e.g.:

```
eclipse -vm /JRE folder/bin/java
```

Invoke the Eclipse executable directly from the directory where it has been installed. For example:

```
<eclipse-install-dir>/eclipse/eclipse
```

3.1.3 Installing on Fedora* Systems

If the Intel C++ Compiler for Linux is installed on an Intel® 64 architecture Fedora* system as a "local" installation, i.e. not installed as root, the installation may fail to properly execute the Eclipse graphical user interfaces of the compiler or debugger. The failure mechanism will typically be displayed as a `JVM Terminated` error. The error condition can also occur if the software is installed from the root account at the system level, but executed by less privileged user accounts.

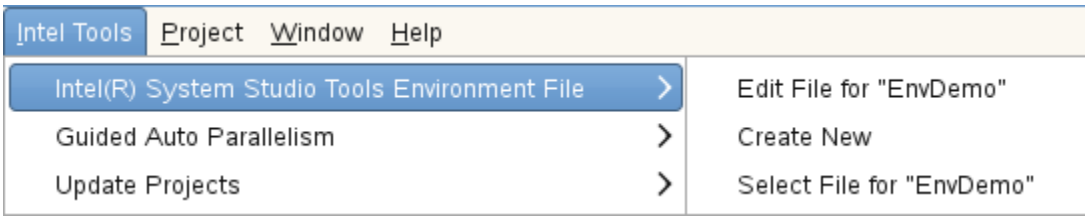
The cause for this failure is that a more granular level of security has been implemented on Fedora, but this new security capability can adversely affect access to system resources, such as dynamic libraries. This new *SELinux* security capability may require adjustment by your system administrator in order for the compiler installation to work for regular users.

3.1.4 Editing Compiler Cross-Build Environment Files

Environment File Support appears under “Intel Tools - Intel® System Studio Tools Environment File” on the menu bar.

If a project has been selected in the Project Explorer, a user will see “Edit File for <project name>, “Create New” and “Select File for <project name>.

If a project has not been selected, a user will see “Edit File” and “Create New”.



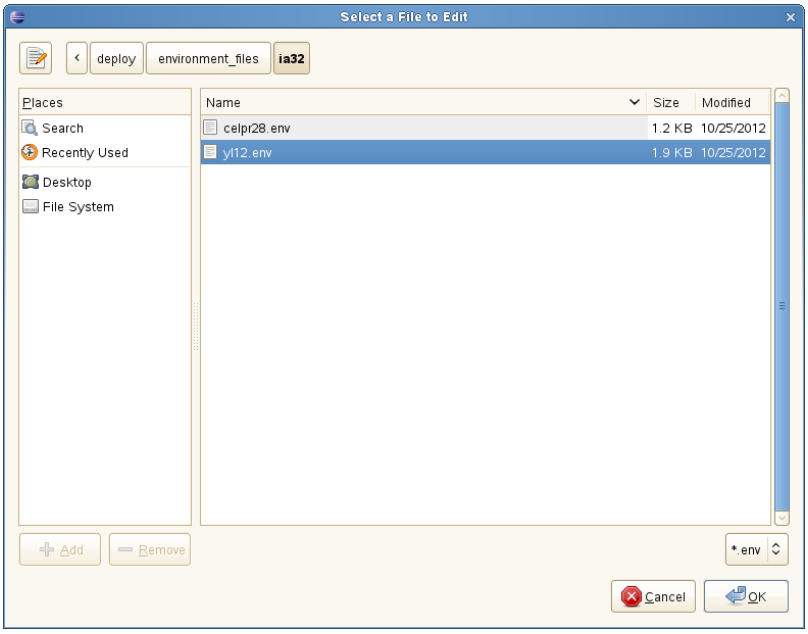
Intel Tools->Intel® System Studio Tools Environment File ->Edit File for <project name>

This option lets a user edit the environment file associated with the current project. If no project is set, an error message box will appear asking them to select a file first.

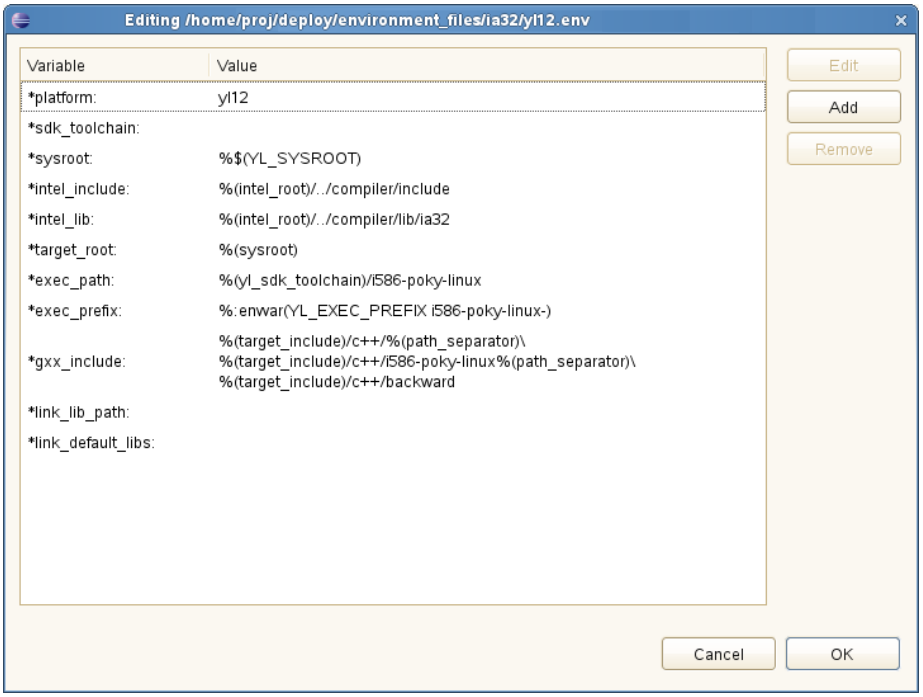
Intel Tools->Intel® System Studio Tools Environment File ->Edit File

on the other hand, lets a user edit any existing environment file.

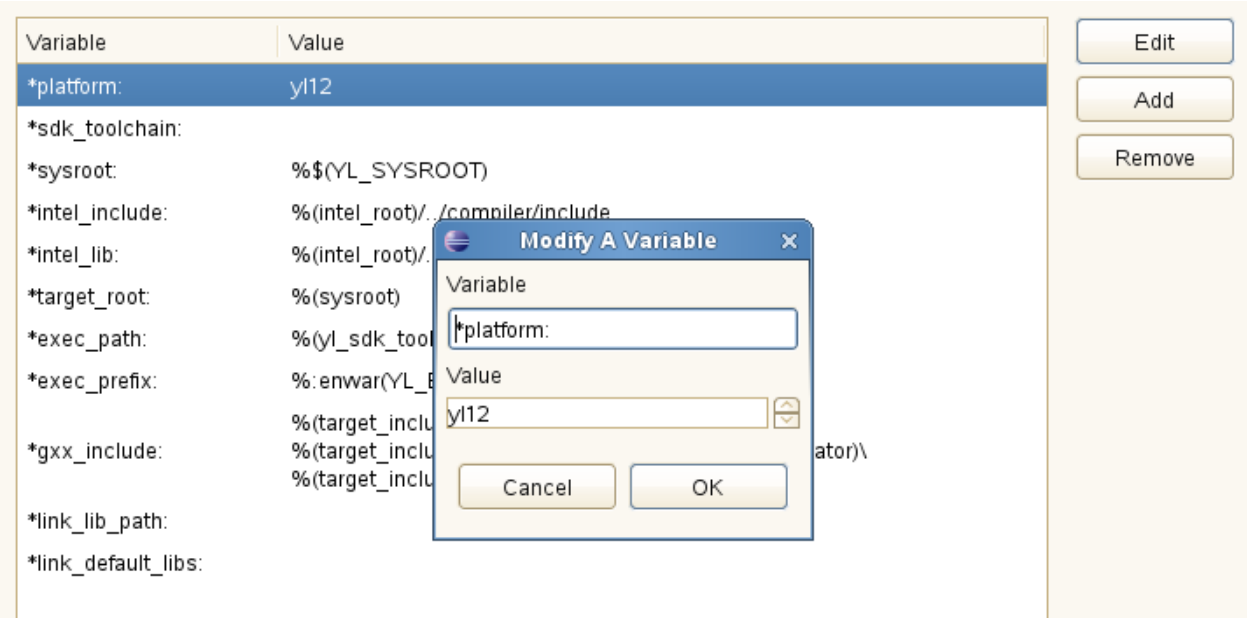
1. In this case, “Select a File to Edit” dialog appears.



2. Then a user will see an edit dialog such as this.



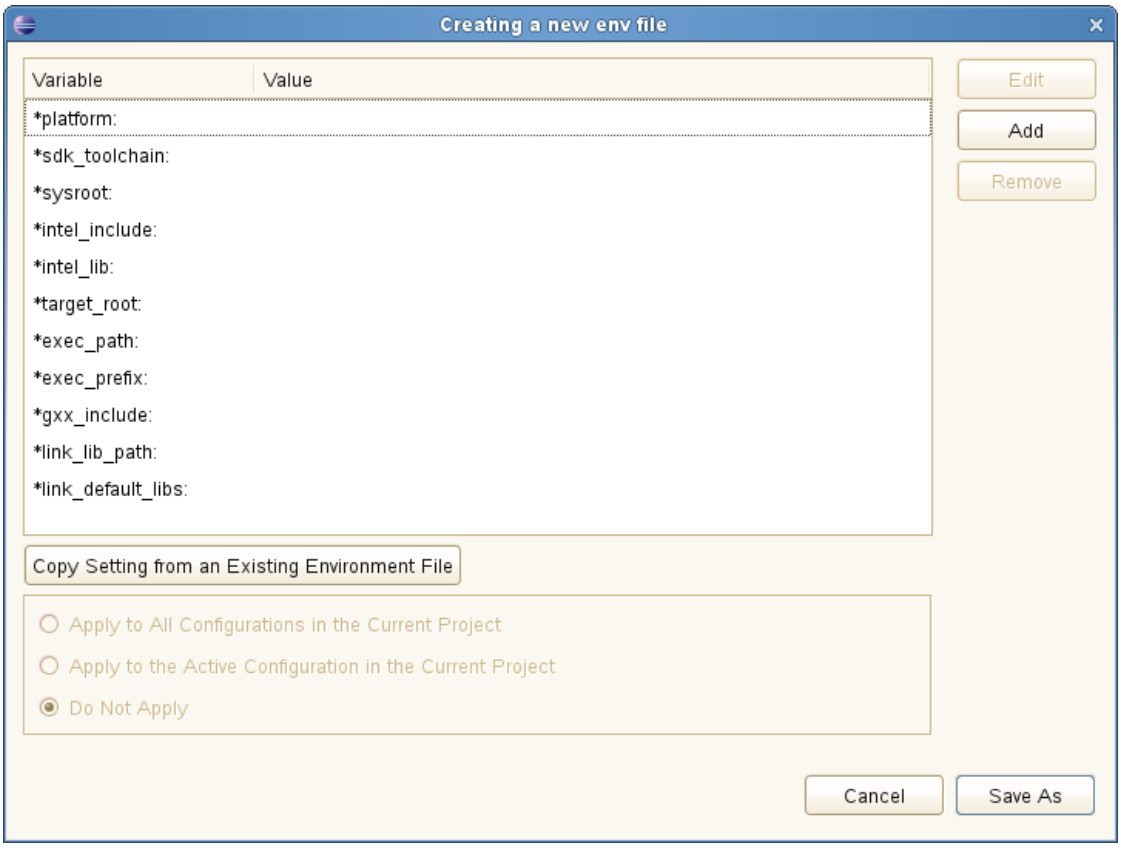
3. To edit a variable value, click on a variable and then “Edit”. “Modify a Variable” box will appear.



4. Select “OK” to save the change and get out of the editing box.
5. Use “Add” to add a new variable-value pair. “Add A New Variable” will appear.
6. To remove a pair, select a pair and select “Remove”. A confirmation dialog will appear. Press OK.
7. When finished with all editing (/ adding / removing), select “OK” in the “Editing..” window.

Intel Tools->Intel® System Studio Tools Environment File ->Create New

This option lets a user create a new .env file. A “Creating a new env file” appears.



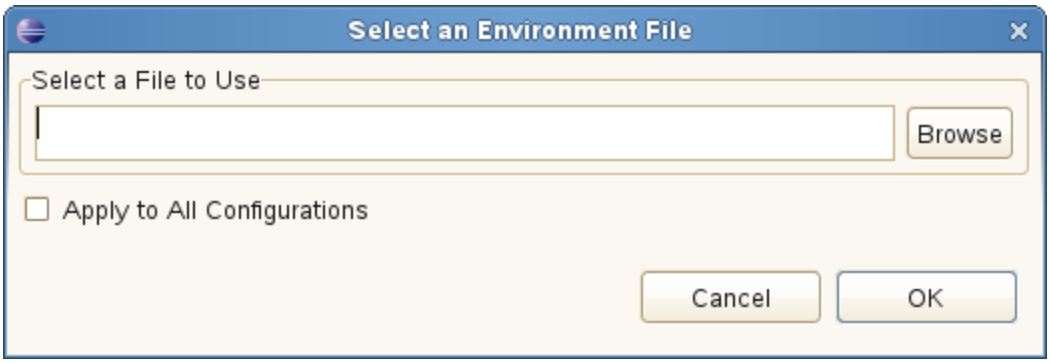
If you'd like to use an existing env file as a base, select "Copy Setting from an Existing Environment File". Then select the base file in the "Select a File to Copy the Setting from" dialog . The values specified in that file now should be propagated to the value fields.

Edit your file.

Select "Apply to All.." – "Do Not Apply" to control where this environment file should be used. If no project has been selected in Project Explorer, these options are grayed out.

Select "Save As" to select the file name and save.

1. Intel Tools->Intel® System Studio Tools Environment File ->Select File for <project name>



This option lets a user select a file that's used with the active configuration the current project. Select "Apply to All Configurations", if the file should be used in all configurations in the current project.

3.1.5 Cheat Sheets

The Intel® C++ Compiler Eclipse* Integration additionally provides Eclipse* style cheat sheets on how to set up a project for embedded use cases using the Intel® C++ Compiler
In the Eclipse* IDE see

Help > Cheat Sheets > Intel C++ Integration > Creating an Intel Embedded Project

Compiler Usage for Cross-Compilation

3.1.6 Build Applications for Yocto Project* Target

The Intel® System Studio installer will automatically integrate the Intel® C++ Compiler with an existing Application Development Toolkit (ADT) installation for Yocto Project* 1.2.x and 1.3

Please refer to the Yocto Project* Application Development Toolkit User's Guide at http://www.yoctoproject.org/docs/current/adt-manual/adt-manual.html for details on the Application Development Toolkit Usage.

To be able to use the Application Development Toolkit with the Intel® C++ Compiler it has to be installed using the Application Development Toolkit tarball and installer script available at http://downloads.yoctoproject.org/releases/yocto/yocto-1.2/adt_installer, http://downloads.yoctoproject.org/releases/yocto/yocto-1.2.1/adt_installer or http://downloads.yoctoproject.org/releases/yocto/yocto-1.3/adt_installer. More detailed information on this can be found at http://www.yoctoproject.org/docs/current/adt-manual/adt-manual.html#using-the-adt-installer

The Yocto Project* distributions and board support packages can be found at http://www.yoctoproject.org/download

Yocto Project* documentation can be found at http://www.yoctoproject.org/documentation

3.1.6.1 Environment Variables

The required environment variables that need to be set using the `source` command are:

```
YL_SYSROOT=/opt/poky/1.2/sysroots/i586-poky-linux
YL_TOOLCHAIN=/opt/poky/1.2/sysroots/i686-pokysdk-linux/usr/bin
```

or

```
YL_SYSROOT=/opt/poky/1.2.1/sysroots/i586-poky-linux
YL_TOOLCHAIN=/opt/poky/1.2.1/sysroots/i686-pokysdk-linux/usr/bin
```

or

```
YL_SYSROOT=/opt/poky/1.3/sysroots/i586-poky-linux
YL_TOOLCHAIN=/opt/poky/1.3/sysroots/i686-pokysdk-linux/usr/bin
```

3.1.6.2 Compiler Platform Option for Cross-Compilation

The compiler option specifying the cross-compilation is

```
icc  -platform=yl12 my_source_file.c
icpc -platform=yl12 my_source_file.c
```

or

```
icc  -platform=yl13 my_source_file.c
icpc -platform=yl13 my_source_file.c
```

The corresponding environment file is located in `<install-dir>/bin/ia32`.

3.1.7 Build Applications for Wind River* Linux* OS Target

To find out more about Wind River* Linux* 5 and on how to install the required Wind River* Linux* toolchain go to http://www.windriver.com/products/linux/

It is based on Yocto Project* and follows many of the same development paradigms http://www.windriver.com/announces/yocto-project/

3.1.7.1 Environment Variables

The required environment variables that need to be set using the `source` command are:

```
WRL_TOOLCHAIN=<install_dir>/wrl50/wrlinux-5/layers/wr-
toolchain/4.6-60
```

```
WRL_SYSROOT=<install_dir>/wrl50/<target>/export/sysroot/intel-
xeon-core_glibc_std/bitbake_build/tmp/sysroots/intel-xeon-core
```

where <install_dir> is the installation path for the Wind River* Linux* software development toolkit, e.g. `/export/SDK` and target is `<ia32>` or `<intel64>`.

3.1.7.2 Compiler Platform Option for Cross-Compilation

The compiler option specifying the cross-compilation is

```
icc  -platform=wrl50 my_source_file.c
icpc -platform=wrl50 my_source_file.c
```

The corresponding environment file is located in `<install-dir>/bin/<target>`

where `<install-dir>` is `/opt/intel/system_studio_2013.0.xxx` and target is `<ia32>` or `<intel64>`.

3.1.8 Build Applications for CE Linux* Target

To get access to the sources and the build environment for your Intel® Atom™ Processor CE42xx or CE53xx based device, please contact the manufacturer of your media streaming device, blu-ray player or IPTV.
If you are a device manufacturer, please contact your Intel Application Engineer to get access.

Below we describe how the Intel® C++ Compiler can interact with a CE Linux* build environment that you installed on your host development machine

3.1.8.1 Environment Variables

The required environment variables that need to be set using the `source` command are:

```
CEL_TOOLCHAIN=<install_dir>/celpr28/staging_dir
CEL_SYSROOT=<install_dir>/celpr28/i686-linux-elf
```

where `<install_dir>` is the installation path for the CE Linux* PR28 product release support package

3.1.8.2 Compiler Platform Option for Cross-Compilation

The compiler option specifying the cross-compilation is

```
icc  -platform=celpr28  my_source_file.c
icpc -platform=celpr28  my_source_file.c
```

The corresponding environment file is located in `<install-dir>/bin/ia32`

where `<install-dir>` is `/opt/intel/system_studio_2013.0.xxx`

3.1.9 Compiler run-time libraries

If you are linking your application built with the Intel® C++ Compiler dynamically, you will need to make the Intel® C++ Compiler runtime libraries available on the target platform. You can do this by following 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. From the thus created directory `../l_cembd_b_2013.0.xxx/rpm/`, copy the file `system_studio_target.tgz`to the target device using ftp, sftp or scp.
3. On the target device unpack the file in a directory to which you have write access
   - `tar -zxvf system_studio_target.tgz`
4. In the thus created directory `../system_studio_target/` the libraries can be found at `../system_studio_target/compiler/lib`.

4 Development Target

The target package system_studio_target.tgz contains Intel® C++ Compiler runtime libraries, the Intel® VTune™ Amplifier Sampling Enabling Product (SEP) , target components for the Intel® VTune™ Amplifier Data Collector, target components for the Intel® Inspector, and the kernel module used by the Intel® JTAG Debugger to export Linux* dynamically kernel module memory load information to host.

To install it 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. From the thus created directory `../l_cembd_b_2013.0.xxx/rpm/`, copy the file `system_studio_target.tgz` to the target device using ftp, sftp or scp.
3. If you used the online installer, the file system_studio_target.tgz can be found in `/tmp/<username>`. Please verify this location in the online installer output.
4. On the target device unpack the file in a directory to which you have write access
   `> tar -zxvf system_studio_target.tgz`
5. Add the compiler runtime libraries that you find in `../system_studio_target/compiler/lib` to your target environment search path.
6. For the dynamic kernel module load export feature follow the instructions found at `../system_studio_target/xdb/kernel-modules/xdbntf/read.me`. This is also detailed in the Intel® JTAG Debugger Installation Guide and Release Notes `jtag-release-install.pdf`.
7. For the Intel® VTune Amplifier Sampling Enabling Product (SEP) follow the instructions in `../system_studio_target/sep_3.10_linux_ia32/docs/SEP_Install_Instructions_Linux.txt`.

4.1.1 Intel® Inspector Command line interface installation

If you would like to install the Intel® Inspector command line interface only for thread checking and memory checking on a development target device, please follow the steps outlined 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. From the thus created directory `../l_cembd_b_2013.0.xxx/rpm/`, copy the file `system_studio_target.tgz`to the target device using ftp, sftp or scp.
3. On the target device unpack the file in a directory to which you have write access
   `> tar -zxvf system_studio_target.tgz`
4. In the thus created directory `../system_studio_target/`, you will find the Intel® Inspector installation at
   `../system_studio_target/inspector_2013_for_systems/`
5. Execute the environment configuration script `inspxe-genvars.sh`.
6. The fully functional command-line Intel® Inspector installation can be found in the `bin32` and `bin64` subdirectories for IA32 and Intel® 64 targets respectively.

4.1.2 Intel® VTune™ Amplifier Command line data collector installation (CLI)

If you would like to install the Intel® VTune™ Amplifier `amplxe-runss` command line data collector for power tuning and performance tuning on a development target device, please follow the steps outlined below:

7. Unpack the tool suite package in a directory to which you have write access.
   `> tar -zxvf l_cembd_p_2013.0.xxx.tgz`
8. From the thus created directory `../l_cembd_p_2013.0.xxx/rpm/`, copy the file `system_studio_target.tgz` to the target device using ftp, sftp or scp.
9. On the target device unpack the file in a directory to which you have write access
   `> tar -zxvf system_studio_target.tgz`
10. In the thus created directory `..system_studio_target/`, you will find the Intel® Amplifier target data collector installation at
    `../system_studio_target/vtune_amplifier_2013_for_systems/target`
11. Data collection on both IA32 and Intel® 64 targets is supported.
12. Follow the instructions in the file `remote-data-collecton.pdf` and `release_notes_amplifier_for_systems.pdf` for more details, on how to use this utility.

4.1.3 Intel® VTune™ Amplifier Sampling Enabling Product

If you would like to install the Intel® VTune™ Amplifier Sampling Enabling Product (SEP), please follow the steps outlined 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. From the thus created directory `../l_cembd_b_2013.0.xxx/rpm/`, copy the file `system_studio_target.tgz`to the target device using ftp, sftp or scp.
3. On the target device unpack the file in a directory to which you have write access
   `> tar -zxvf system_studio_target.tgz`
4. In the thus created directory `../system_studio_target/`, you will find the Intel® VTune Amplifier Sampling Enabling Product at
   `../system_studio_target/sep_3.10_axeu3_linux_x32` or
   `../system_studio_target/sep_3.10_axeu3_linux_x64`
5. Follow the instructions in
   `../system_studio_target/sep_3.10_linux_<target>/docs/SEP_User_Guide.pdf` and
   `../system_studio_target/sep_3.10_linux_<target>/README.TXT`

4.1.4 Using Intel® VTune™ Amplifier with Virtualization

1. Dynamic binary instrumentation based data collection for hotspot analysis, locks&waits analysis and concurrency analysis have been tested with VMWare* 5.1.
2. Event based Sampling and Sampling with stacks data collection is functional inside a Virtual machine only for architectural events.
   Please see the Performance Monitoring Unit Sharing Guide at http://software.intel.com/file/30388/ for more details on the supported architectural events
   VMWare* 5.1 and KVM* 1.0.50 have been verified to support this.
3. Power and Frequency data collection is not supported in a virtualized environment.

4.1.5 Intel® Integrated Performance Primitives redistributable shared object installation

If you are using dynamic linking when using the Intel® Integrated Performance Primitives, you will need to copy the relevant Linux* shared objects to the target device along with the application.
A list of the redistributable shared objects can be found at
`/opt/intel/system_studio_2013.0.xxx/documentation/ippredist.txt`

4.1.6 Intel® Math Kernel Library redistributable shared object installation

If you are using dynamic linking when using the Intel® Math Kernel Libraries, you will need to copy the relevant Linux* shared objects to the target device along with the application.
A list of the redistributable shared objects can be found at
`/opt/intel/system_studio_2013.0.xxx/documentation/mklredist.txt`

5 Debug

Selecting the provided GDB

To use the provided GNU* Project Debugger GDB instead of the default GDB debugger provided with the default GNU* tools installation of your distribution, please source the following debugger environment setup script:

```
<install-dir>/system_studio_2013.0.xxx/debugger/gdb/bin/
debuggervars.sh
```

Integrating the provided GDB into Eclipse* for remote debug

Remote debugging with GDB using the Eclipse* IDE requires installation of the C/C++ Development Toolkit (CDT) (http://www.eclipse.org/downloads/packages/eclipse-ide-cc-linux-developers-includes-incubating-components/indigosr2) as well as Remote System Explorer (RSE) plugins (http://download.eclipse.org/tm/downloads/). In addition RSE has to be configured from within Eclipse* to establish connection with the target hardware.

1. Copy the `gdbserver` provided by the product installation (`<install-dir>/system_studio_2013.0.xxx/debugger/gdb/<arch>/<python>/bin/`) to the target system and add it to the execution PATH environment variable on the target.

2. Configure Eclipse* to point to the correct GDB installation:

   a. Inside the Eclipse* IDE click on Window>Preferences from the pulldown menu.

   b. Once the preferences dialogue appears select C++>Debug>GDB from the treeview on the left.

   c. The GDB executable can be chosen by editing the “GDB debugger” text box. Point to `<install-dir>/system_studio_2013.0.xxx/debugger/gdb/<arch>/<python>/bin/`,

      where `<arch>` is ia32 or intel64 and `<python>` is py24, py26, or py27, depending on architecture and Python* installation

Using GDB to debug application on embedded devices

Please refer to GDB: The GNU* Project Debugger (http://www.gnu.org/software/gdb/) for details.

For cross-development GDB comes with a remote debug agent called gdbserver. This debug agent can be installed on the debug target to launch a debuggee and attach to it remotely from the development host.

This can be useful in situations where the program needs to be run on a target host that is different from the host used for development, particularly when the target has a limited amount of resources (either CPU and/or memory).

To do so, start your program using gdbserver on the target machine. gdbserver then automatically suspends the execution of your program at its entry point, waiting for a debugger to connect to it. The following commands start an application and tells gdbserver to wait for a connection with the debugger on localhost port 2000.

```
$ gdbserver localhost:2000 program
Process program created; pid = 5685
Listening on port 2000
```

Once gdbserver has started listening, we can tell the debugger to establish a connection with this gdbserver, and then start the same debugging session as if the program was being debugged on the same host, directly under the control of GDB.

```
$ gdb program
(gdb) target remote targethost:4444
Remote debugging using targethost:4444
0x00007f29936d0af0 in ?? () from /lib64/ld-linux-x86-64.so.
(gdb) b foo.adb:3
Breakpoint 1 at 0x401f0c: file foo.adb, line 3.
(gdb) continue
Continuing.

Breakpoint 1, foo () at foo.adb:4
4       end foo;
```

It is also possible to use gdbserver to attach to an already running program, in which case the execution of that program is simply suspended until the connection between the debugger and gdbserver is established. The syntax would be

```
$ gdbserver localhost:2000 --attach 5685
```

to tell gdbserver to wait for GDB to attempt a debug conenction to the running process with process ID 5685

Using GDB to debug applications running inside a virtual machine

Using GDB for remotely debugging an application running inside a virtual machine follows the same principle as remote debug using the gdbserver debug agent.

The only additional step is to ensure TCP/IP communication forwarding from inside the virtual machine and making the ip address of the virtual machine along with the port used for debug communication visible to the network as a whole.

Details on how to do this setup can be found on Wikibooks* (http://en.wikibooks.org/wiki/QEMU/Networking)

The basic steps are as follows

1. Install QEMU, the KQEMU accellerator and bridge-utils

   ```
   $ su -

   $ yum install qemu bridge-utils
   ```

2. Creating the image for the guest OS

   For best performance, you should install your guest OS to an image file. To create one, type:

   ```
   $ qemu-img create filename size[ M | G ]
   ```

   where **filename** is going to be the name of your image, and **size** is the size of your image with the suffix 'M' (MB) or 'G' (GB) right after the number, no spaces.

   ```
   $ qemu-img create Linux.img 10G
   ```

3. Configuring network for your guest OS

   Put the following contents into **/etc/qemu-ifup**:

   ```
   #!/bin/sh
   #
   # script to bring up the device in QEMU in bridged mode
   #
   # This script bridges eth0 and tap0. First take eth0 down, then bring it up with IP 0.0.0.0
   #
   /sbin/ifdown eth0
   /sbin/ifconfig eth0 0.0.0.0 up
   #
   # Bring up tap0 with IP 0.0.0.0, create bridge br0 and add interfaces eth0 and tap0
   #
   /sbin/ifconfig tap0 0.0.0.0 promisc up
   /usr/sbin/brctl addbr br0
   ```

```
/usr/sbin/brctl addif br0 eth0
/usr/sbin/brctl addif br0 tap0
#
# As we have only a single bridge and loops are not possible, turn spanning tree protocol off
#
/usr/sbin/brctl stp br0 off
#
# Bring up the bridge with IP 192.168.1.2 and add the default route
#
/sbin/ifconfig br0 192.168.1.2 up
/sbin/route add default gw 192.168.1.1
#stop firewalls
/sbin/service firestarter stop
/sbin/service iptables stop
```

The bold values can be changed. Please change the IP's to show your setup - The first bold is a comment, so it doesn't really matter. The second bolded value is the IP identical to the one assigned to your computer (means you'll need static IPs so you can predict your IP) and the third and last is your gateway.

Now, put this into /etc/qemu-ifdown:

```
#!/bin/sh
#
# Script to bring down and delete bridge br0 when QEMU exits
#
# Bring down eth0 and br0
#
/sbin/ifdown eth0
/sbin/ifdown br0
/sbin/ifconfig br0 down
#
# Delete the bridge
#
/usr/sbin/brctl delbr br0
#
# bring up eth0 in "normal" mode
#
/sbin/ifup eth0
#start firewalls again
/sbin/service firestarter start
/sbin/service iptables start
```

Make the scripts executable so QEMU can use them:

```
$ su -

$ chmod +x /etc/qemu-if*

$ exit
```

4. Installing the guest OS

   Type the following to start the installation:

   ```
   $ su

   $ /sbin/modprobe tun

   $ qemu -boot d -hda image.img -localtime -net nic -net tap -m 192
   -usb -soundhw sb16 -cdrom /dev/hdc;/etc/qemu-ifdown
   ```

   Where **image.img** was the name you gave to your image earlier. I'm also assuming /dev/cdrom is your CD drive - if it's not, then please change it to the correct device. After the install is complete, proceed to step 5.

5. Making the run script & running at will

   The last step is to create the QEMU start script and from there on you can run your guest OS. Create this file - called **qemustart** - in the same directory as your image:

   ```
   #!/bin/sh

   su -c "/sbin/modprobe tun;qemu -boot c -hda image.img -localtime
   -net nic -net tap -m 192 -usb -soundhw sb16;/etc/qemu-ifdown"
   ```

   Where **image.img** was the name given to the image earlier.
   Last step - make the startup script executable:

   ```
   $ chmod +x /path/to/qemustart
   ```

Intel® JTAG Debugger

Please refer to the Getting Started Guide

<install-dir>/intel_studio_2013.0.xxx/
documentation/en_US/debugger/xdb/first_use/Getting_Started.htm

and the Intel® JTAG Debugger release notes

<install-dir>/intel_studio_2013.0.xxx/documentation/en_US/debugger/xdb/jtag-release-install.pdf

6 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.

7 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.