Workshop Setup Guide

Introduction

This document contains information for setting up an Ubuntu 12.04 host computer to run the lab exercises of the “Introduction to Embedded Linux One-Day Workshop.”

It consists of 4 required sections:

- Installing Ubuntu 12.04
- Installing Code Composer Studio
- Installing Lab Files
- Configuring Ubuntu Static IP

After completing these installation steps, you will have everything needed to run the lab exercises on your system.

Additionally, a number of steps were taken to make the environment more user friendly (“Installing Gnome3 and Standard Scrollbars”) and to set up the lab files toolchain and target filesystem. These comprise the three optional sections. There is no need to go through the optional sections in order to run the lab exercises, but if you would like to know the steps that were required to set up that portion of the lab environment, the steps are shown in these optional sections.

The lab files that you will need to install are located on the workshop wiki page at:

http://processors.wiki.ti.com/index.php/Introduction_to_Linux_One-Day_Workshop

Chapter Topics

Installing Ubuntu 12.04

Note: Currently the Eclipse IDE platform upon which Code Composer Studio is based uses 32-bit libraries. It is recommended that you install the 32-bit desktop Ubuntu 12.04 even if you have a 64-bit machine.

There are many tutorials available for installing Ubuntu, so this section will not go through great detail on the actual installation; however, it provides information for removing the user password and setting automatic login, as is done in the workshop image.

1. **Begin by downloading an Ubuntu 12.04 image and burning onto an installation disk.**
   Please see above. It is recommended that you install the 32-bit version of Ubuntu even if you are using a 64-bit machine. There are tools that allow installation via an installation CD (requires a CD burner to create the installation disk) as well as installation via a USB mass-storage device. Either method is fine.
2. **Install Ubuntu 12.04 on your computer.**
   Other versions of Ubuntu may also work, but version 12.04 is what has been tested for this workshop.
   If you select “automatic login” on the user setup screen, you can skip step 4.
   Be sure to write down the password that you set! The following steps will show you how to remove the password, but you will need to know the old one.
3. **Open a terminal**
   ctrl-alt-t
4. **Select automatic login (If you forgot to select in step 2)**
   ```
   # sudo gedit /etc/lightdm/lightdm.conf
   ```
   Enter the password from step 2 when prompted.
   Add the following four lines under the section header “[SeatDefaults]”
   ```
   autologin-guest=false
   autologin-user=user
   autologin-user-timeout=0
   autologin-session=lightdm-autologin
   ```
5. **Allow null passwords for sudo**
   ```
   # sudo gedit /etc/sudoers
   ```
   Enter the password from step 2 if prompted.
   Locate the line that reads:
   ```
   %admin ALL=(ALL) ALL
   ```
   And change to read
   ```
   %admin ALL=(ALL) NOPASSWD: ALL
   ```
6. **Allow null passwords for authorization (i.e. login)**
   ```
   # sudo gedit /etc/pam.d/common-auth
   ```
   Locate the line that contains “nullok_secure” and change “nullok_secure” into just “nullok”
7. **Remove user password**
   ```
   # sudo passwd –d user
   ```
8. **Reboot to test**
   ```
   # sudo shutdown –h now
   When Ubuntu reboots, open a terminal and try the sudo command:
   # sudo ls
   ```
   If everything has worked correctly, the list operation should complete without prompting for a password.

Installing Code Composer Studio v.6.0.1.00040

You can download CCS6 from:

`http://processors.wiki.ti.com/index.php/CCSv6`

and selecting the “Linux” download button.

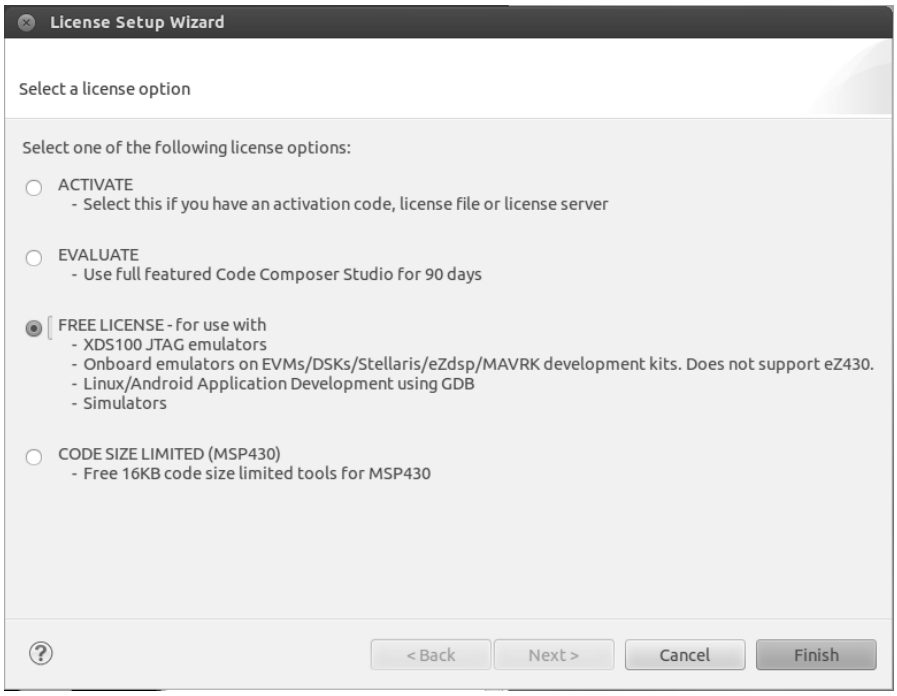
Also, because older versions of CCS are not always archived, the exact version used in the workshop is also available from the wiki page.

9. **Install required libraries**
   ubuntu# sudo apt-get update
   ubuntu# sudo apt-get install libgnomevfs2-0
   ubuntu# sudo apt-get install liborbit2-dev
10. **Download CCS from the above listed link**
11. **Add executable permission to the installer**
    `# chmod a+x ccs_setup_linux32.bin`
12. **Run the installer program**
    `# ./ccs_setup_linux32.bin`
13. **Read and accept the license agreement.**
14. **Accept the default install directory: /home/user/ti**
15. **Select “Sitara 32-bit ARM Processors” from the Processor Support window, and in the submenu select “GCC ARM Compiler”**
    You should not select “TI ARM Compiler.” This compiler does not have Linux support and is used when Sitara processors are run without the Linux operating system. The “GCC Arm Compiler” will not actually be used to generate code because OpenEmbedded will be used instead, but this compiler is used to parse source files.
16. **In “Select Emulators” window, leave the defaults of XDS100 and XDS200 support.**
    These emulators are supported under the free license.
17. **Leave the default for “App Center” (no packages installed)**
    Though PRU and Linux Development Tools are interesting packages for AM335x development, they will not be used in this workshop.
18. **Press “Finish” to begin the installation**
19. **Select “Create Desktop Shortcut” on final screen (Should be selected by default.)**
20. **Change desktop shortcut to a custom script**
    Right-click the shortcut and select properties
    Change “command:”
        From: /home/user/ti/ccsv6/eclipse/ccstudio
        To: /home/user/ti/ccsv6/eclipse/my_ccstudio.sh
21. **Create “myccstudio.sh” launch script**
    # gedit /home/user/ti/ccsv6/eclipse/my_ccstudio.sh

    Enter the following:

    export PATH=/home/user/gcc-linaro-arm-linux-gnueabihf-4.7-2013-03-20130313_linux/bin:$PATH
    /home/user/ti/ccsv6/eclipse/ccstudio

22. **Make the script executable**
    # chmod a+x /home/user/ti/ccsv6/eclipse/my_ccstudio.sh
23. **Start Code Composer Studio**
24. **Select the Free License Agreement**
    The License Agreement selection will come up upon first start.



If for some reason the License Setup Wizard does not automatically launch, you can access it via:
Help→Code Composer Studio Licenscing Information→Upgrade Tab→Launch Licensing Setup…

25. **Close CCS**

Installing Workshop Files

The merged Yocto and Open Embedded projects provide an open-source set of recipe files for rebuilding various distributions using a tool named Bitbake. The generic Yocto/OE installation is one route to rebuilding Arago, but it is recommended that you instead install the Yocto/OE build environment from the arago-project.org website as it has been pre-configured for Arago builds.

1. **Use aptitude to install the sg3-utils package (used in lab 0)**
   ```
   ubuntu$ sudo apt-get install sg3-utils
   ```
1. **Use aptitude to install qvfb (used in lab 10)**
   ```
   ubuntu# sudo apt-get install qt4-dev-tools
   ```
1. **Install bitbake dependencies**
   ```
   ubuntu$ sudo apt-get install git build-essential diffstat texinfo
   gawk chrpath python-dev python-m2crypto
   ```
2. **Switch shell from dash to bash**
   ```
   ubuntu$ sudo dpkg-reconfigure dash
   ```
   select “no” when prompted
3. **Install Linaro cross compiler**
   ```
   ubuntu$ cd /home/user
   ubuntu$ wget --no-check-certificate \
      https://launchpad.net/linaro-toolchain-
   binaries/trunk/2013.03/+download/gcc-linaro-arm-linux-gnueabihf-
   4.7-2013.03-20130313_linux.tar.bz2
   ```

   ubuntu$ tar -jxvf gcc-linaro-arm-linux-gnueabihf-4.7-2013.03-20130313_linux.tar.bz2 -C /home/user
4. **Add Linaro path to .bashrc**
   ubuntu$ gedit /home/user/.bashrc

   Add the following at the end:
   export PATH=$PATH:/home/user/gcc-linaro-arm-linux-gnueabihf-4.7-2013.03-20130313_linux/bin

5. **Source .bashrc for change to take effect**
   ubuntu# source /home/user/.bashrc
6. **Install arago bitbake files**
   Ubuntu$ git clone git://arago-project.org/git/projects/oe-layersetup.git
7. **Configure for sdk-07.01.00.00**
   Ubuntu$ cd oe-layersetup
   Ubuntu$ ./oe-layertool-setup.sh –f configs/amsdk/amsdk-07.01.00.00-config.txt
8. **Add workshop overlay**
   ubuntu$ cd /home/user/oe-layersetup/sources
   ubuntu$ git clone https://github.com/preissig/meta-workshop
9. **Add meta-workshop layer to bitbake configuration**
   ubuntu$ cd /home/user/oe-layersetup/
   ubuntu$ gedit build/conf/bblayers.conf

   Add “/home/user/oe-layersetup/sources/meta-workshop” to the “BBLAYERS” variable.
10. **(Optional) Get preloaded packages**
    Over time the locations and versions of packages change on the internet. Generally you will have the best success, especially with older builds, by downloading one of the preloaded source code packages. This is also significantly faster than having bitbake download each file individually.

```
Ubuntu$ wget
http://downloads.ti.com/dsps/dsps_public_sw/am_bu/sdk-
downloads/TISDK-Downloads/ALL/exports/amsdk-07.01.00.00-
downloads.tar.gz
Ubuntu$ tar zxf amsdk-07.01.00.00-downloads.tar.gz
Ubuntu$ mv sdk-7.1-downloads/* oe-layersetup/downloads
Ubuntu$ rmdir sdk-7.1-downloads
```

11. **Build sdk – <u>this will take a long time, possibly 24 hours</u>!**
    Ubuntu$ cd build
    Ubuntu$ source conf/setenv
    Ubuntu$ MACHINE=am335x-evm bitbake -k arago-core-tisdk-image
    (Note: the "-k" option will continue to build the distribution even if an error is encountered in one of the packages.)
    For available images:
    # ls sources/meta-arago/meta-arago-distro/recipes-core/images/
12. **Copy sdk image**
    ubuntu$ cp arago-tmp-external-linaro-toolchain/deploy/images/arago-core-tisdk-image-am335x-evm.tar.gz /home/user
13. **De-archive sdk**
    Ubuntu$ mkdir ti-sdk-07.01.00.00
    Ubuntu$ tar zxf arago-core-tisdk-image-am335x-evm.tar.gz –C /home/user/ti-sdk-07.01.00.00
14. **Run sdk installer**
    ubuntu$ ./sdk-install.sh

15. **Run sdk set up**

```
ubuntu# sudo apt-get update
ubuntu$ sudo ./setup.sh
```

When prompted, enter username as "user" instead of the default of "root."

Accept default values for each other question until you get to the serial port for minicom. Instead of the default of "/dev/ttyS0" enter "/dev/ttyUSB1"

You may then accept the default values for the rest of the script.

Configuring Ubuntu Static IP

The “auto” setting for usb0 in /etc/network/interfaces is a workaround. It would be better specified as “allow-hotplug” however, there are known issues with this in Ubuntu 12.04. The web recommends using udev as an alternate solution, but the workshop developer was unable to make this approach work.

Using “auto usb0” works well, but with the disadvantage that if no ethernet-over-usb connection is available when Ubuntu starts up, the message “waiting on network configuration…” will appear and will require about 2 minutes to timeout. This extra 2 minutes of boot time may be circumvented by attaching the beaglebone so that the interface is present.

Users who dislike this 2 minute boot time may remove “auto usb0” in which case the usb0 will have to be manually configured each time the Beaglebone is attached using “#sudo ifup usb0”

2. **Open /etc/network/interfaces file**
   # sudo gedit /etc/network/interfaces
3. **Add an “eth0” entry (or modify current entry.) Entry should be as follows:**

```
auto eth0
iface eth0 inet static
      address 192.168.1.1
      netmask 255.255.255.0

auto usb0
iface usb0 inet static
      address 192.168.7.1
      netmask 255.255.255.0
```

   Note: “address” and “netmask” entries preceded by tab.
4. **Save and close**
5. **Remove Gnome networking settings**

```
# sudo nm-connection-editor
```

   Any connection that appears under the “wired” or “wireless” tab should be deleted.
6. **(Optional) Reboot and use “ifconfig” to verify new setting**

```
# ifconfig
```

7. **Open /etc/hosts**

```
# sudo gedit /etc/hosts
```

8. **Add static IP addresses for hosts on the network**
   (At the end of the file, add the following)

```
192.168.1.1 ubuntu.gigether.net
192.168.1.2 am335x.gigether.net
192.168.7.1 ubuntu.etherusb.net
192.168.7.2 am335x.etherusb.net
```

(Optional) Installing Gnome3 and Standard Scrollbars

Ubuntu 12.04 ships with a desktop manager called Unity. One feature that a lot of people do not prefer in Unity is that the drop-down lists that would normally appear at the top of a window (including CCS) now appear at the top of the desktop. Additionally, Unity uses a new type of scrollbar called overlay scrollbars that, while saving a little space on the screen that can be used for other things, are a little more difficult to use.

This section is not required for the workshop labs to work properly, but since these changes were made on the workshop image, they are listed here.

1. **Launch a terminal**
2. **Acquire a WAN (i.e. internet) connection**
   If you have already set up a static IP address as per the previous section, you can override the static address using
   ```
   ubuntu$ sudo ifdown eth0
   ubuntu$ sudo dhclient eth0
   ```
3. **Install gnome-shell Aptitude package**
   ```
   ubuntu$ sudo add-apt-repository ppa:gnome3-team/gnome3
   ubuntu$ sudo apt-get update
   ubuntu$ sudo apt-get install gnome-shell
   ```
4. **Log out of the Ubuntu session**
   There is a gear icon in the top right corner that produces a drop-down menu with the logout option.
5. **Select the Gnome Desktop**
   Click the Ubuntu icon next to thye username (user) and select Gnome.
6. **Press login to log back in**
   The desktop has only subtly changed, but if you launch CCS, you will notice that the pulldown menus are now at the top of the CCS window (instead of along the top of the desktop.)
7. **Disable overylay scrollbars**
   Launch a terminal and type the following (single line, no carriage return)
   ```
   ubuntu$ gsettings set org.gnome.desktop.interface ubuntu-overlay-scrollbars false
   ```
8. **Log out and back in for change to take effect**
   The “user” dropdown menu in the top right of the desktop can be used to log out.