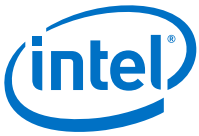
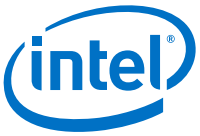
Enabling Hardware Accelerated Video Decode on Intel® Atom™ Processor D2000 and N2000 Series under Fedora 16

Application Note

October 2012

Order Number: 509577-003US





Contents

1.0 Introduction 5

2.0 Overview of Hardware Accelerated Decode 5

3.0 Build and Installation Overview 6

4.0 Step-by-Step Instructions 7

4.1 Cedar Trail Platform Preparation 7
4.2 Fedora 16 Installation on Cedar Trail Target Platform 7
4.3 Build Kernel on Host and Install on Target 8
4.4 Build Kernel on Target and Install 11
4.5 Install Packages on Target 14
4.6 Compile MPlayer on Target 14

5.0 Play Video and Check CPU Utilization 15

6.0 Run Glxgears and Check CPU Utilization 15

7.0 References 15

8.0 Acronyms 16

Figures

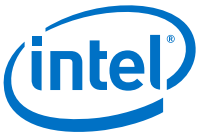
1 Overview of Hardware Accelerated Decode 6
2 Menu Config Utility Screen 9
3 Menu config Utility - Intel CDV Driver selection 10
4 Menu Config Utility Screen 12
5 Menu Config Utility - Intel CDV Driver selection 13

Tables

1 Acronym Table 16

Revision History

Date	Revision	Description
October 2012	003	Updated download center link.
October 2012	002	Updated patch information for graphics drivers.
July 2012	001	Initial release



1.0 Introduction

The Cedar Trail platform is based on the Intel® Atom™ Processor N2000 and D2000 Series processor (formerly called Cedarview) and the Intel® NM10 Express Chipset (formerly called Tiger Point). This processor series is based on a 32nm process and features new levels of performance-per-watt opening the door to always-on, always-connected embedded devices.

The N2000/D2000 series processors include an integrated graphics controller with advanced media handling capabilities such as smooth full HD (up to 1080p) video playback along with support for wide range of outputs such as VGA, LVDS, HDMI*, DP* and eDP[1]. Media playback is optimum when the video decoding is handled by the video engine in the integrated graphics controller. Using the video engine can significantly reduce the CPU workload and also help improve the quality of playback.

Intel has released graphics and media drivers (referred to as PVR-CDV drivers in this document) for the MeeGo open-source distribution for the Cedar Trail platform. The PVR-CDV drivers exploit the video and graphics acceleration capabilities of the media engine. This application note is a case study on integrating these drivers with the Fedora distribution and then demonstrating the video acceleration capability of the system with the MPlayer video player. The steps to demonstrate graphics acceleration using the glxgears demo utility are also provided.

The case study was done on a system with Intel® Atom N2800 processor and Intel® NM10 Express Chipset. However, the findings are applicable in general to any Cedar Trail system.

2.0 Overview of Hardware Accelerated Decode

To meet the demands of low power and high performance, the Cedarview processor provides dedicated graphics and video decode acceleration hardware for delivering fast video and graphics rendering. Hardware acceleration frees up most of the CPU bandwidth for other time critical tasks. In addition to lowering CPU utilization, hardware acceleration also helps improve the quality of media playback by reducing or eliminating frame drop and reducing audio/video synchronization issues. Figure 1 on page 6 provides a simplified overview of the graphics and video acceleration stack.

VA-API is a standard API that exposes offloading of video decoding to acceleration hardware. `libVA` is an open source library implementation of the VA-API specification. This library provides access to the hardware used for acceleration of video processing. It enables hardware accelerated video decode at various entry points (VLD, IDCT, Motion Compensation, deblocking) for the prevailing coding standards today (MPEG-2, MPEG-4 ASP/H.263, MPEG-4 AVC/H.264, and VC-1/WMV3). The VA hardware driver is the Cedarview hardware-specific video decode driver.

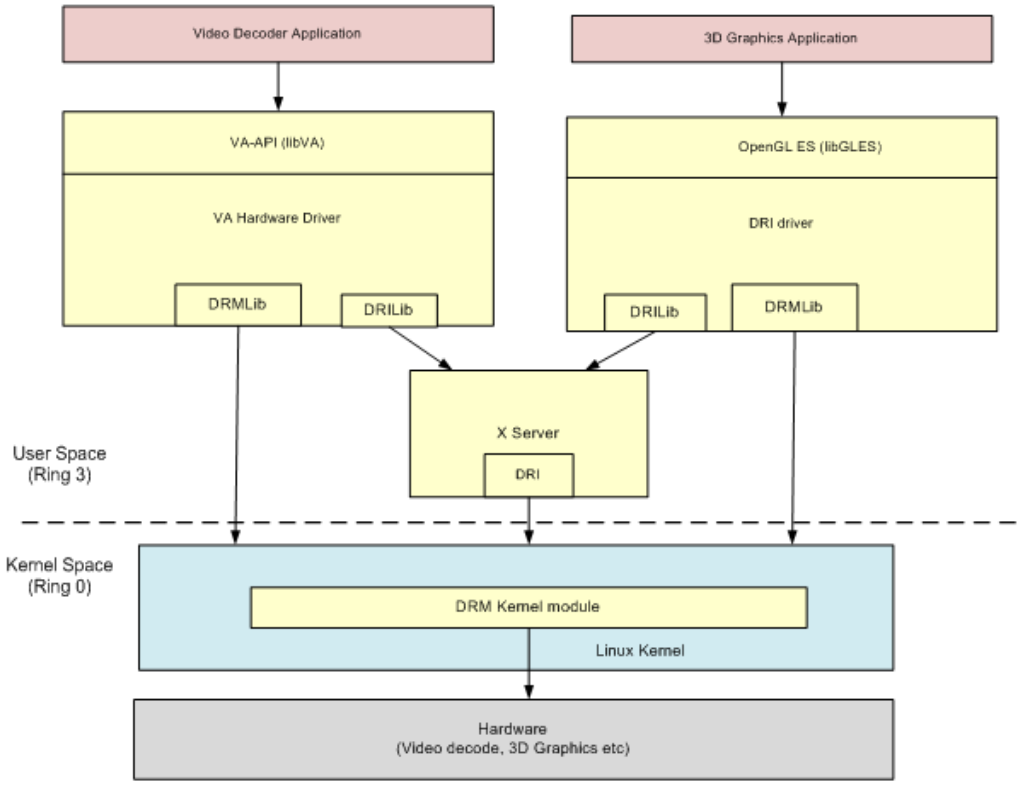
OpenGL ES API is used to expose 3D graphics acceleration to the graphics application[2]. libGLES is the open source library implementation of the OpenGL ES API. The DRI driver converts libGLES commands to the graphics hardware accelerator-specific commands.

The X server provides the basic services for managing windows displays and input devices. It provides the basic framework for building GUIs.

1. Version 1.0 of PVR-CDV drivers does not support DP and eDP ports; support was added in v.1.0.1.
2. OpenGL API is partially supported by PVR-CDV drivers. However, OpenGL ES is the recommended API for 3D graphics applications.

DRM (Direct Rendering Manager) kernel module is the kernel space portion of the "Direct Rendering Infrastructure" (DRI). The DRM kernel module provides synchronized access to the graphics controller from the video and graphics drivers and the X server.

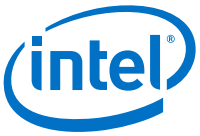
Figure 1. Overview of Hardware Accelerated Decode



3.0 Build and Installation Overview

The build and installation of the Cedar Trail software stack is done as follows:

1. Build and install the Cedar Trail target kernel with required patches. The target kernel is built by patching the 3.1.0 kernel release from kernel.org with PVR-CDV kernel patches. Note that the target kernel can be natively built either on the target itself or cross-compiled on a host machine. This document provides instructions for both cases.
2. Install drivers, libraries, and other packages needed for hardware accelerated playback. The graphics driver used in this case study is from the MeeGo v1.2 release for Cedar Trail platform. The driver has the following dependencies.
   - Mesa GL 7.11
   - X Server 1.11
   - Kernel 3.1.0
   - Patched libwsbm 1.1.0+ (provided in release)
   - llibc >= 2.11.90
   - libva >= 1.0.15



- pixman >= 0.22.0
- gcc >= 4.5.1
- libdrm >= 2.4.25

3. Build and install MPlayer on the target machine. The MPlayer is patched to use VA-API based hardware acceleration for video decoding instead of using the CPU for video decode.

4.0 Step-by-Step Instructions

4.1 Cedar Trail Platform Preparation

1. Ensure that the board has DDR memory installed (minimum of 1 GB recommended).
2. Connect a VGA or HDMI monitor to the board.
   This study was done with a VGA and HDMI monitor, however other displays supported by the Cedar Trail platform and driver can also be used such as: LVDS, DP or eDP.
3. Connect a USB keyboard and a mouse to the board.
4. Connect a hard disk drive to the board.
5. Connect a DVD player to the board using a SATA cable.
6. Connect the power supply to the board.

4.2 Fedora 16 Installation on Cedar Trail Target Platform

Follow the steps below to install the Linux Fedora 16, 32-bit (i686) OS on the Cedar Trail target board.

1. Disconnect the LAN cable during the installation to prevent Fedora 16from going online and updating the packages and causing the wrong kernel version to be installed on the system.
2. Turn ON the power supply connected to the target board.
3. Power on the board. (On most systems this would typically require momentarily pushing the Power-On button.)
4. You should now see the boot menu on the VGA/HDMI monitor as seen below:
   “No boot device has been detected, please press any key to reboot!”
5. Insert the Fedora 16 i686 DVD in the DVD player.
6. Press **Enter**. The Fedora OS installation begins on the hard disk drive.
7. Select **Install new system** or **Upgrade** from the menu. Select **Test the Media** before installation.
8. When the Fedora 16 menu appears, click **Next**.
9. Select **English** as the installation language.
10. Select **U.S.English** for the keyboard.
11. Select **Basic Storage Devices** and then click **Next**.
12. Select **Fresh Installation** to start a new installation (note that the old data will be lost).
13. Use the default (**localhost.localdomain**) host name and then click **Next**.
14. Select your time zone and then click **Next**.

15. Type the root password, store it in a safe place and then click **Next**.
16. Select **Replace existing Linux System** and then click **Next**.
17. Select **Write Changes to Disk**.
18. Select **Software Development**, **Installation Repo**, **Customize Now**, and then click **Next**.

Note: Selecting **Software Development** allows the right tools to be installed without using `yum` later that may install updated packages that we don't want right now.

19. Under Desktop Environments, select **Xfce**, and deselect GNOME because the F16 uses GNOME 3 and that requires OpenGL. The Cedarview driver provides support for OpenGLES 2.0, but limited “big” OpenGL support.
20. After Installation is complete, remove the DVD from the player and select **Reboot**.
21. Reconnect the LAN cable.
22. After reboot, you can see the welcome menu. Click **Forward**.
23. On the License Information screen, click **Forward**.
24. Create a user name and password and then click **Forward**.
25. Set the appropriate date and time in the calendar and then click **Forward**.
26. Click **Finish**.
27. Log in and select **Default** on the desktop option.
28. Disable `SELinux`:
    vi /etc/selinux/config
    SELINUX=disabled

Note: SELINUX will not be disabled until AFTER the system is rebooted.

29. Enable `eth0` to activate the Internet connection using Network Manager, which appears as a button or a network symbol on the top-right of screen.

4.3 Build Kernel on Host and Install on Target

Follow these steps only if you intend to cross-compile the target kernel by building it on a host machine. Cross-compiling can result in significantly faster builds than compilation on the Cedar Trail target itself. The build time is approximately 3 hours. The steps assume that Fedora 16 is installed in the host along with packages required for compilation. If your host does not have a Fedora 16 installed, then you can follow steps similar to Section 4.2 to install it on the host, except that you can use either 32-bit or 64-bit Fedora 14 Linux.

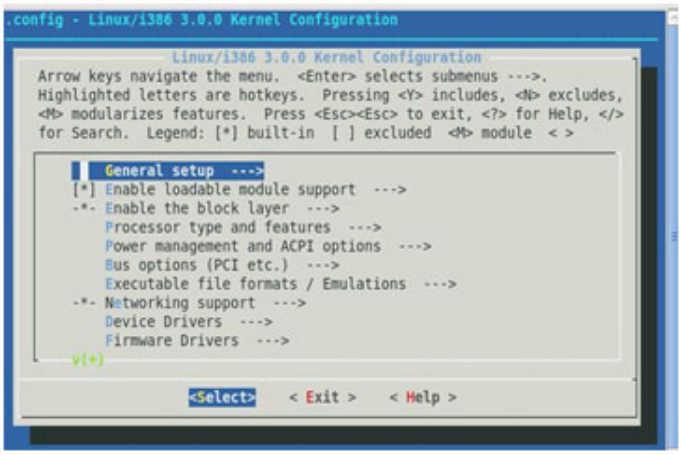
Use the following steps to build the kernel to be installed on the Cedar Trail target board.

1. Go to the website:
   http://www.kernel.org/pub/linux/kernel/v3.x/
2. Look for the following file and download it:
   `linux-3.1.tar.gz`
3. Download the patch tarball (`cdv-gfx-drivers-1.0.3_bee.tar.bz2`) from the following location:
   http://downloadcenter.intel.com/Detail_Desc.aspx?agr=Y&DwnldID=21938
4. Extract the contents of the patch in home directory:
   tar xjvf cdv-gfx-drivers-1.0.3_bee.tar.bz2

5. Untar the kernel in home directory:
   tar xzf linux-3.1.tar.gz

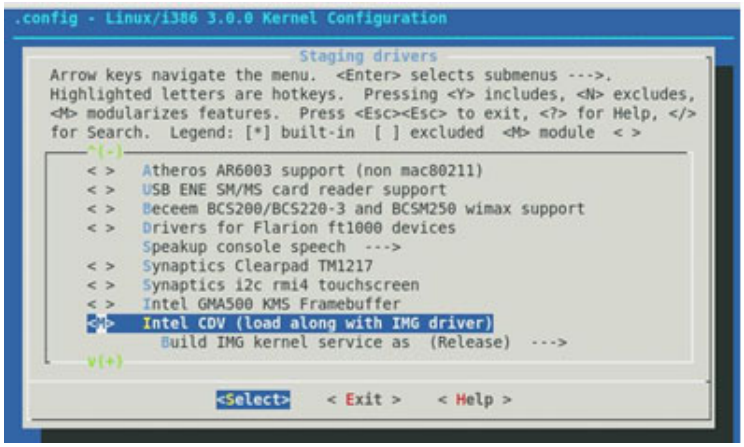
   This command creates a directory named `linux-3.1` and untars all the files into this directory.
6. Apply the kernel patches by using the patch utility:
   cd linux-3.1
   patch -p1 < ../cdv-gfx-drivers-1.0.3_bee/src/kernel-ttm-clear-high.patch
   patch -p1 < ../cdv-gfx-drivers-1.0.3_bee/cedarview-kernel-v1.0.3_bee.patch
7. Edit `drivers/staging/Kconfig` and add the following line:
   source "drivers/staging/cdv/Kconfig"
8. Edit `drivers/staging/Makefile` and add the following line:
   obj-$(CONFIG_DRM_INTEL_CDV) += cdv/
9. We are now ready to build the kernel. Note that the next step will use the config file from the current installations /boot directory which could be a PAE image. We will want to make sure that we reset that option with `menuconfig`.
10. Configure and compile the kernel:
    linux32 make menuconfig
    The kernel configuration window pops up and looks like the following:

Figure 2. Menu Config Utility Screen

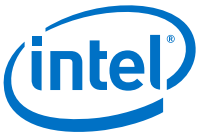


a. Using the arrow key and **Enter**, select **Processor type and features --- >**
b. Using the arrow key and **Enter**, select **High Memory Support (64GB) --->**
c. If you see **High Memory Support (4GB) -->** instead, skip the next 2 steps.
d. Move the cursor to 4 **GB** and press the spacebar to select.
e. Exit one level.
f. Using the arrow key and **Enter**, select **Device driver --- >**
g. Using the arrow key and **Enter**, select **Staging drivers --- >**
h. Select **Intel CDV (load along with IMG driver)** and then press the spacebar. This will load the release driver.

Figure 3. Menu config Utility - Intel CDV Driver selection



11. Exit and save.
12. Make rpm to load into the target:
    linux32 make rpm
13. During build process the `rpmbuild` directory is created in your home directory and there are four directories created under `rpmbuild`.
    a. **cd ~/rpmbuild/RPMS/i386**
    b. copy the following two files to a USB drive
       `kernel-3.1.0-1.i386.rpm`
       `kernel-headers-3.1.0-1.i386.rpm`
14. Install the new kernel on the target system:
    a. Login as root on the target system.
    b. Create a `cedartrail` directory on the target:
       mkdir cedartrail
    c. Copy the following files from the USB to the `cedartrail` directory:
       `kernel-3.1.0-1.i386.rpm`
       `kernel-headers-3.1.0-1.i386.rpm`
    d. Install the kernel:
       rpm -ivh kernel-3.1.0-1.i386.rpm
       rpm -ivh kernel-headers-3.1.0-1.i386.rpm
       Now you should see a new directory called `3.1.0` under `/lib/module`.
15. Make the ramdisk:
    mkinitrd /boot/initramfs-3.1.0.img 3.1.0
16. Add the new kernel to the grub files:
    grub2-mkconfig -o /boot/grub2/grub.cfg



17. Edit `/etc/default/grub` as follows:
    a. If your display is not LVDS, in the kernel line, add:
    `video=LVDS-1:d`

    This is a workaround to an issue that causes the driver to believe that an LVDS display is connected.
    b. To enable video media playback to work correctly, in the kernel line, add `vmalloc=256MB`
    c. Enter the following command:
    grub2-mkconfig -o /boot/grub2/grub.cfg
18. Unpack and install `cedarview-libwsbm` binary tarball relative to your root file system. Assume patch tarball was extracted in `/home/jim`:
    cd /
    tar xvf /home/jim/cdv-gfx-drivers-1.0.3_bee/bin/cedarview-libwsbm-1.1.0.tar
19. Install libva and dependencies from your package manager:
    yum -y install libva
20. Unpack and install the appropriate user space (Xorg, 2D, 3D) Cedarview drivers relative to your root file system. While the PowerVR driver is a Mesa replacement for EGL and GL ES, `mesa-libGL` of the stated version is required for OpenGL operation. Note that development headers are installed as well.
    tar xvf /home/jim/cdv-gfx-drivers-1.0.3_bee/bin/cedarview-userspace-v1.0.3_bee.tar
21. Unpack the PVR VA-API driver for Cedartrail-accelerated H.264, MPEG-2, and VC-1 streams:
    tar xvf /home/jim/cdv-gfx-drivers-1.0.3_bee/bin cedarview-vaapi-v1.0.3_bee.tar
22. Reboot and make sure you select the correct new image when grub displays the list of choices. You can now use the following command to set the default:
    grub2-set-default <title or number>

4.4 Build Kernel on Target and Install

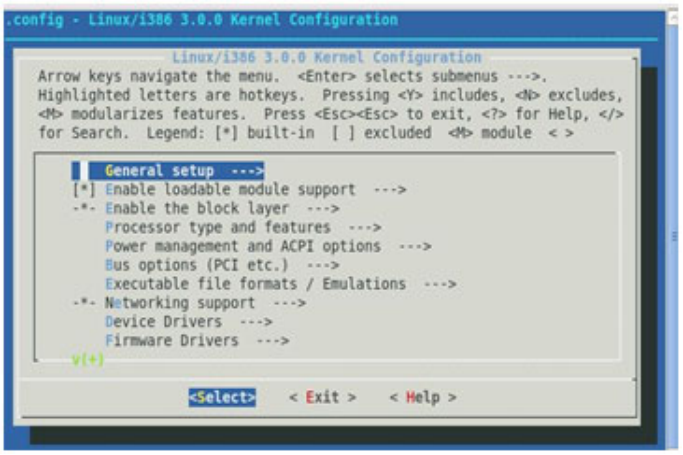
Follow these steps only if you intend to natively build the kernel to be installed on the Cedar Trail target board itself. The build time is approximately 3 hours. Now we have a hard disk with Fedora 16 Xfce installed for the target based on the steps described in Section 4.2.

1. Go to the website:
   http://www.kernel.org/pub/linux/kernel/v3.x/
2. Look for the following file and download it:
   `linux-3.1.tar.gz`
3. Download the patch tarball from downloadcenter.intel.com (`cdv-gfx-drivers-1.0.3_bee.tar.bz2`)
   http://downloadcenter.intel.com/Detail_Desc.aspx?agr=Y&DwnldID=21938
4. Extract the contents of the patch in home directory:
   tar xjvf cdv-gfx-drivers-1.0.3_bee.tar.bz2
5. Untar the kernel in home directory:
   tar xzf linux-3.1.tar.gz
   This command creates a directory named `linux-3.1` and untars all the files into this directory.

6. Apply the kernel patches by using the patch utility:
   cd linux-3.1
   patch -p1 < ../cdv-gfx-drivers-1.0.3_bee/src/kernel-ttm-clear-high.patch
   patch -p1 < ../cdv-gfx-drivers-1.0.3_bee/cedarview-kernel-v1.0.3_bee.patch
7. Edit `drivers/staging/Kconfig` and add the following line:
   `source "drivers/staging/cdv/Kconfig"`
8. Edit `drivers/staging/Makefile` and add the following line:
   obj-$(CONFIG_DRM_INTEL_CDV) += cdv/
9. We are now ready to build the kernel. Note that the next step will use the config file from the current installations `/boot` directory which is a PAE image. We will want to make sure that we reset that option with `menuconfig`.
10. Configure and compile the kernel:
    make menuconfig

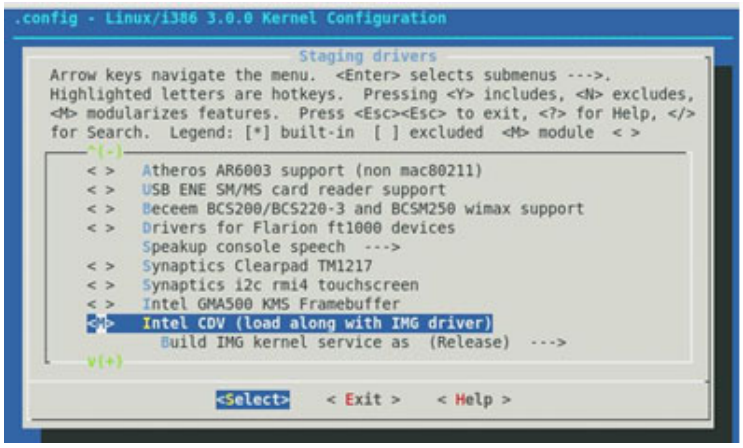
    The kernel configuration window pops-up and looks like the following:

Figure 4. Menu Config Utility Screen



a. Using the arrow key and **Enter**, select **Processor type and features --- >**
b. Using the arrow key and **Enter**, select **High Memory Support (64GB) --->**
c. Move the cursor to 4 **GB** and press the spacebar to select.
d. Exit one level.
e. Using the arrow key and **Enter**, select **Device driver --- >**
f. Using the arrow key and **Enter**, select **Staging drivers --- >**
g. Select **Intel CDV (load along with IMG driver)** and then press the spacebar. This will load the release driver.

Figure 5. Menu Config Utility - Intel CDV Driver selection



11. Exit and save.
12. Make and install the kernel

    make

    su -c "make modules_install"

    su -c "make install"

    If the make is successful, the kernel should be there. Check that ramdisk was made in the `/boot` directory.
13. Unpack and install the `cedarview-libwsbm` binary tarball relative to your root file system. Assume the patch tarball was extracted in `/home/jim`:

    cd /

    tar xvf /home/jim/cdv-gfx-drivers-1.0.3_bee/bin/cedarview-libwsbm-1.1.0.tar
14. Install libva and dependencies from your package manager:

    yum -y install libva
15. Unpack and install the appropriate user space (Xorg, 2D, 3D) Cedarview drivers relative to your root file system. While the PowerVR driver is a Mesa replacement for EGL and GL ES, `mesa-libGL` of the stated version is required for OpenGL operation. Note that development headers are installed as well.

    tar xvf /home/jim/cdv-gfx-drivers-1.0.3_bee/bin/cedarview-userspace-v1.0.3_bee.tar
16. Unpack the PVR VA-API driver for Cedartrail-accelerated H.264, MPEG-2, and VC-1 streams:

    tar xvf /home/jim/cdv-gfx-drivers-1.0.3_bee/bin cedarview-vaapi-v1.0.3_bee.tar
17. Edit `/etc/default/grub` as follows:

    a. If your display is not LVDS, in the kernel line, add:
    `video=LVDS-1:d`

    This is a workaround to an issue that causes the driver to believe that an LVDS display is connected.

    b. In the kernel line, add
    `vmalloc=256MB`

18. Update the `grub.cfg` so our changes take effect:
    grub2-mkconfig -o /boot/grub2/grub.cfg
19. Reboot and make sure you select the correct new image when grub displays the list of choices. You can now use the following command to set the default:
    grub2-set-default <title or number>

4.5 Install Packages on Target

There are several dependencies for the graphics driver and MPlayer. The steps below show how to install the required packages on the target system. The assumption is that the initial installation was done with the **Software Development** option.

Install Xorg packages as root:

yum -y install libXi-devel libXfont-devel libXext-devel
yum -y install libdrm-devel
yum -y install libpciaccess-devel pixman-devel
yum -y install expat-devel
yum -y install libva-devel freeglut-devel

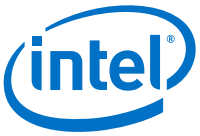
4.6 Compile MPlayer on Target

Use these steps to build MPlayer with hardware accelerated video decoding in the target system. Before proceeding, ensure that the system has an Internet connection.

1. Install yasm and git:
   yum -y install yasm git
2. Install some alsa dependencies:
   yum -y install alsa*
3. Clone MPlayer git repository:
   git clone git://gitorious.org/vaapi/mplayer.git

   If the command executes successfully, the MPlayer tree is downloaded in the target under the `mplayer` directory.
4. Configure MPlayer:
   cd mplayer
   git checkout -t origin/hwaccel-vaapi
   ./configure

   At the **No FFmpeg** checkout prompt, press **Enter**.
   cd ffmpeg
   git checkout -b ffmpeg-0.6.3 [you can use the latest version tag]
5. Get out of `ffmpeg` directory:
   cd ..
6. Now we are in the MPlayer directory. Configure MPlayer with options as follows:
   ./configure --prefix=/usr/local --enable-xv --enable-gl --enable-vaapi --disable-vdpau --disable-mencoder --disable-faad --enable-runtime-cpudetection
7. Compile MPlayer:
   make
   make install



5.0 Play Video and Check CPU Utilization

1. Ensure that MPlayer is built with `vaapi` enabled:
   mplayer -vo help

   `vaapi` should be among the list of video output drivers listed by the above command.
2. Play the video with MPlayer using hardware acceleration:
   mplayer -vo vaapi -va vaapi -ao alsa <example_video_clip>.mp4
3. To check CPU utilization, open another terminal and enter:
   top
   Max CPU utilization for the MPlayer application should be about 10% (with 1080p videos).

6.0 Run Glxgears and Check CPU Utilization

1. To check whether graphics hardware acceleration is enabled, enter the following commands:
   glxinfo | grep renderer

   The string `OpenGL renderer string: PowerVR SGX545` should appear in the output, indicating that the PVR-CDV OpenGL driver has been correctly installed.

   glxinfo|grep direct

   In the output, look for the string `direct rendering: Yes`, which indicates that graphics hardware acceleration is enabled.
2. Launch the glxgears demo:
   glxgears -fullscreen

   A full screen animation of three rotating gears starts to play.
   Press the **Esc** key to stop glxgears.
3. Launch the `glxgears` demo (non-full screen):
   glxgears

   The glxgears animation starts in a new window.
4. To check that CPU utilization, open another terminal and enter:
   top

   Maximum CPU utilization for the `glxgears` application should be less than 2%.

7.0 References

1. VA-API:
   http://www.freedesktop.org/wiki/Software/vaapi
2. MeeGo. The MeeGo project site hosts the Cedar Trail graphics drivers and kernel patches (Released as part of the Cedar Trail Netbook platform). Refer to this site to check for any updates:
   https://meego.com/

8.0 Acronyms

Table 1. Acronym Table

API	Application Programming Interface
AVC	Audio Video Coding
DP	DisplayPort
eDP	Embedded Display Port
HDMI	High Definition Multimedia Interface
IDCT	Inverse Discrete Cosine Transform
LVDS	Low Voltage Differential Signaling
HD	High Definition
VA-API	Video Acceleration API
VGA	Video Graphics Array
VLD	Variable Length Decoding
WMV3	FourCC notation for Windows Media Video 9

§ §