By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

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.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL’S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS’ FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

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

Any software source code reprinted in this document is furnished for informational purposes only and may only be used or copied and no license, express or implied, by estoppel or otherwise, to any of the reprinted source code is granted by this document.

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%5Fnumber/

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

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

Copyright © 2014, Intel Corporation. All rights reserved.
Contents

1.0 Introduction .................................................................................................................. 7
  1.1 Purpose .................................................................................................................... 7
  1.2 Intended Audience ................................................................................................. 7
  1.3 Related Documents ............................................................................................... 8
  1.4 Conventions .......................................................................................................... 8
  1.5 Acronyms and Terminology .................................................................................. 8

2.0 Architectural Overview ................................................................................................. 11
  2.1 Introduction ............................................................................................................. 11
    2.1.1 Display Options ............................................................................................... 12
  2.2 Features .................................................................................................................... 13
    2.2.1 Processor(s) Supported .................................................................................. 13
    2.2.2 OS Support ..................................................................................................... 13

3.0 Configuration and CED Lite .......................................................................................... 15
  3.1 Environment Setup ................................................................................................. 15
  3.2 Creating a Configuration in CED Lite – Summary Steps ........................................ 16
  3.3 Display Setup ......................................................................................................... 16
  3.4 Entering Advanced Settings ................................................................................. 17
  3.5 Entering Pipe Settings ............................................................................................ 18
  3.6 Setting Splash Screen Options ............................................................................. 19
  3.7 Setting Splash Video Options ................................................................................ 20
    3.7.1 Configuring Splash Video ............................................................................... 21
    3.7.2 How to Select the Video_Offset ...................................................................... 22
  3.8 CRTC Settings ........................................................................................................ 22
  3.9 Importing a DTD .................................................................................................... 23
  3.10 Creating a New Customized DTD ......................................................................... 23
  3.11 DTD Example Specifications ............................................................................... 26
  3.12 Sprite Plane Settings ............................................................................................. 26
  3.13 Configuring HDMI, DP, eDP and VGA Ports ...................................................... 27
  3.14 Configuring eDP or VGA Flat Panel Settings ..................................................... 29
  3.15 Setting HDMI, DP, eDP, and VGA Framebuffer Color Correction Attributes .. 29
  3.16 Configuring HDMI, DP, eDP, or VGA Attribute Settings .................................... 30
  3.17 Port Driver Attributes ......................................................................................... 30
    3.17.1 Standard Port Driver Attributes .................................................................. 30
    3.17.2 HDMI Port Driver Attributes ...................................................................... 31
    3.17.3 Default Search Order ................................................................................... 31
    3.17.4 Default GPIO Pin Pair Assignments ............................................................. 31
  3.18 Generating an Intel® EMGD Configuration ............................................................ 32
  3.19 Options .................................................................................................................. 32
  3.20 Reset ..................................................................................................................... 32
  3.21 New xorg options in EMGD SNA: ........................................................................ 32
    3.21.1 Plane Reservation xorg configuration: ......................................................... 32
    3.21.2 XvAdapter xorg configuration ..................................................................... 33

4.0 Firmware: GOP, EPOG, and VBIOS .......................................................................... 35
  4.1 Overview .................................................................................................................. 35
  4.2 System Requirements for Video Firmware ............................................................ 35
  4.3 Embedded Pre-OS Graphics (EPOG) Driver ......................................................... 35
    4.3.1 EPOG Overview ............................................................................................ 35
    4.3.2 System Requirements .................................................................................... 36
    4.3.3 Configuring EPOG through CED Lite ............................................................ 36
5.0 Installing Linux* OS Drivers ..................................................................................43
  5.1 Overview ...............................................................................................................43
  5.2 Pre-requisites, Supported Hardware and Software Requirements for Intel®
      EMGD ...................................................................................................................44
      5.2.1 Supported Hardware ..................................................................................44
  5.3 Installation of Intel® EMGD on a Fedora* Platform ...........................................44
  5.4 Configuring and Installing the User Space Libraries on Fedora* .......................46
      5.4.1 Installing Dependencies for User Space Libraries ..................................46
      5.4.2 Installing the Direct Rendering Manager (DRM) Library .......................46
      5.4.3 Installing 3-D Components ......................................................................47
      5.4.4 Configuring Accelerated Video Encode and Decode for Intel® EMGD ......48
  5.5 Configuring the Intel® EMGD on Linux* OS Using CED Lite and Generating
      Binary Blob ...........................................................................................................56
      5.5.1 Linux* OS Configuration Using CED Lite ...............................................56
      5.5.2 Prerequisites for CED Lite ........................................................................58
      5.5.3 Creating the Configuration .......................................................................58
      5.5.4 Generating and Installing the Binary Blob .............................................60
      5.5.5 Generating the Configuration Files with the Firmware Binary Blob ..........61
  5.6 The Linux* OS Configuration File .......................................................................62
      5.6.1 Device Section ...........................................................................................62
      5.6.2 Screen Section ...........................................................................................63
      5.6.3 ServerLayout Section ................................................................................63
  5.7 Runtime Configuration GUI ...............................................................................63
      5.7.1 Emgdgui Setup ..........................................................................................64
      5.7.2 Using the emgdgui Runtime Configuration Utility ..................................64
  5.8 Hue Control Support for Sprite ..........................................................................70
  5.9 Xrandr .................................................................................................................70
  5.10 Intel® EMGD Custom Control Using the sysfs Interface ..................................71
  5.11 Sprite Performance through VAAPI Stack with VAXV Support .......................73
      5.11.1 Enable Sprite in VAAPI Stack with VAXV Support ..............................73
      5.11.2 Tools to Test the Feature ........................................................................73
  5.12 Flipping an OGL Surface onto Sprite Using Xprop ............................................74
  5.13 Enabling HDMI Audio ......................................................................................74
  5.14 Enabling DisplayPort Audio .............................................................................75
  5.15 RC6 Feature .......................................................................................................75
  5.16 EMGD S3D Modes .............................................................................................76
  5.17 Enabling Hot-plug .............................................................................................77

A Fedora* 18 Bring-up with Kernel 3.8 on Intel® E3800/N2807/N2930/J1900 ...........79
  A.1 Prerequisites ......................................................................................................79
  A.2 Steps ..................................................................................................................79
  A.3 Installing Live Bootable Image into the Hard Disk Drive .................................79
  A.4 Update Fedora* 18 to Latest Kernel Version ..................................................86

B 2D/3D API Support .................................................................................................89
  B.1 2D Support .......................................................................................................89
  B.2 3D Support .......................................................................................................89
      B.2.1 OpenGL* APIs ........................................................................................90
      B.2.2 OpenGL* ES 1.1 ....................................................................................94
      B.2.3 OpenGL* ES 2.0 ....................................................................................94
      B.2.4 EGL .....................................................................................................95

C Example User_config.c File ...............................................................................97
# Figures
1. Intel® Embedded Media and Graphics Driver ............................................................... 11
2. Splash Video with 8 MB of Stolen Memory Example ...................................................... 21
3. Example Runtime Configuration GUI - Driver Info Tab ................................................... 65
4. Example Runtime Configuration GUI - CRTC/Pipes Tab .............................................. 66
5. Example Runtime Configuration GUI - Planes Tab ..................................................... 67
6. Example Runtime Configuration GUI - Modeset Tab ................................................... 68
7. Example Linux* Runtime Configuration GUI - Color Correction Tab ............................ 69

# Tables
1. Acronyms and Terminology ......................................................................................... 8
2. Types of Displays Supported ........................................................................................ 12
3. Display Configuration Definitions ............................................................................... 12
4. Supported Display Configurations ................................................................................ 12
5. Processors Supported by Intel® EMGD v. 36.40.23 and Intel® EMGD v. 37.40.23 for Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processors N2807/N2930/J1900 Linux* Release ........................................................ 13
6. Linux* OS Settings .................................................................................................... 17
7. Pipe Settings ............................................................................................................. 18
8. Splash Screen Options ............................................................................................... 19
9. Splash Video Options ................................................................................................. 20
10. Primary CRT Settings ............................................................................................... 22
11. Intel® EMGD DTD Setting Options ........................................................................... 24
12. Timing Specification Example Values ......................................................................... 26
13. Color Correction Values (applies to ALL color) .......................................................... 26
14. Port Settings ........................................................................................................... 27
15. Flat Panel Settings Options ........................................................................................ 29
16. Linux* Framebuffer Color Correction Values (applies to R, G, B color) ....................... 29
17. Standard Port Driver Attributes .............................................................................. 30
18. Default Search Order ............................................................................................... 31
19. Default GPIO Pin Pair Assignments ......................................................................... 31
20. "S3D Set" Connector Property ................................................................................... 76
21. "expose 3D modes" Connector Property .................................................................... 77
22. Intel-Supported OpenGL® APIs for Linux* ................................................................ 90
23. Non-Supported OpenGL® APIs ................................................................................ 93
24. Non-Supported OpenGL ES APIs ............................................................................ 95
## Revision History

This document may have been updated since the release shown below. See [http://edc.intel.com/Software/Downloads/](http://edc.intel.com/Software/Downloads/) for the most recent version.

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>August 2013</td>
<td>001</td>
<td>Initial release for Intel® Embedded Media and Graphics Driver and EPOG v36.40.1 (32-bit) &amp; v37.40.1 (64-bit) for Bay Trail SoC Linux* Release</td>
</tr>
<tr>
<td>October 2013</td>
<td>002</td>
<td>Hot fix release for Initial release for Intel® Embedded Media and Graphics Driver and EPOG v36.40.1 (32-bit) &amp; v37.40.1 (64-bit) for Bay Trail SoC Linux*</td>
</tr>
<tr>
<td>December 2013</td>
<td>003</td>
<td>Intel® Embedded Media and Graphics Driver and EPOG v36.40.21 (32-bit) and v37.40.21 (64-bit) for Bay Trail SoC Linux* Release</td>
</tr>
<tr>
<td>August 2014</td>
<td>004</td>
<td>Intel® Embedded Media and Graphics Driver and EPOG v36.40.23 (32-bit) and v37.40.23 (64-bit) for Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processor N2807/N2930/J1900 Linux* Release</td>
</tr>
</tbody>
</table>
1.0 Introduction

The Intel® Embedded Media and Graphics Driver (Intel® EMGD) graphics driver suite is designed to meet the requirements of embedded applications. Featuring Intel® Dynamic Display Configuration Technology (Intel® DDCT), the drivers run on the following Embedded Intel® Architecture (eIA) chipset:

- Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processor N2807/N2930/J1900 (Linux*)

Note: If you need support for a chipset that is not listed above but is in the same family as those listed, first check for an appropriate version for download on the Intel® Embedded Design Center (http://www.intel.com/p/en_US/embedded/hwsw/software/emgd#download) or contact your Intel representative. Intel does not support other variations of video firmware that are not stated in the driver package release notes or this user guide.

The Intel® EMGD supports the following types of display devices:

- Analog CRT
- HDMI
- DisplayPort
- Embedded DisplayPort (eDP)

Intel® EMGD is designed to work with fixed-function systems, such as Point-of-Sale (POS) devices, ATMs, gaming devices, In-vehicle Information/Entertainment systems, etc. It can be configured to work with various hardware and software systems and supports Linux® operating systems, including embedded versions of these operating systems.

Note: Refer to product release notes for more details on Intel® EMGD supported features.

1.1 Purpose

This help system provides information on both firmware and software, including hardware design considerations, installation requirements, and static configuration options.

1.2 Intended Audience

This help system targets all platform and system developers who need to interface with the graphics subsystem. This includes, but is not limited to: platform designers, system BIOS developers, system integrators, original equipment manufacturers, system control application developers, and end users.
1.3 Related Documents

The following documents provide additional information that can be useful when using the Intel® Embedded Media and Graphics Driver. Additional resources are available at http://www.intel.com/go/EMGD (http://edc.intel.com/Software/Downloads/EMGD/).

- **Intel® Graphics Media Accelerator - Binary Modification Program (BMP) User’s Guide Revision 3.0 or above (Document Number: 368119).**
  This document provides information on configuration of Video BIOS.

- **Intel® HD Graphics Media Accelerator - Binary Modification Program (BMP) User’s Guide for GOP VBT Revision 0.9 or above (Document Number: 488363).**
  This document provides information on configuration of GOP.

- **VESA BIOS Extensions/Display Data Channel Standard**
  This document provides information on the 4F VBE functions, which are supported by the Intel® Embedded Video BIOS.

- **VESA BIOS Extension (VBE) Core Functions Standard Version 3.0**
  Contains information on the VESA BIOS Extension (VBE) specification for standard software access to graphics display controllers that support resolutions, color depths, and framebuffer organizations beyond the VGA hardware standard.

*Note:* The last two documents above are available from http://www.vesa.org. Membership may be required to access these documents. Reproductions may also be available from elsewhere on the Internet.

1.4 Conventions

The following conventions are used throughout this document:

<table>
<thead>
<tr>
<th><strong>Boldface</strong></th>
<th>Represents text that you type and text that appears on a screen.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Italics</strong></td>
<td>Introduces new terms and titles of documents.</td>
</tr>
<tr>
<td><strong>Courier New</strong></td>
<td>Identifies the names of files, executable program names, and text that appears in a file.</td>
</tr>
<tr>
<td><strong>Angle Brackets (&lt;&gt;)</strong></td>
<td>Encloses variable values in syntax or value ranges that you must replace with actual values.</td>
</tr>
</tbody>
</table>

1.5 Acronyms and Terminology

The table below lists the acronyms and terminology used throughout this document:

<table>
<thead>
<tr>
<th>Table 1. Acronyms and Terminology (Sheet 1 of 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Term</strong></td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>API</td>
</tr>
<tr>
<td>BDA</td>
</tr>
<tr>
<td>BIOS</td>
</tr>
<tr>
<td>CED Lite</td>
</tr>
<tr>
<td>Clone Display Configuration</td>
</tr>
</tbody>
</table>
### Introduction—Intel® EMGD

**Contrast Ratio**
Contrast ratio is the measure of the difference between light and dark on a display. If the contrast is increased, the difference between light and dark is increased (i.e., something white will be very bright and something black will be very dark). Brightness and Contrast Controls differ in function between CRTs and LCDs.

**DC**
Display Configuration.

**DP**
Display Port.

**DTD**
Detailed Timing Descriptor. A set of timing values used for EDID-less devices.

**EBDA**
Extended BIOS Data Area. An interface that allows the system BIOS and Option ROMs to request access to additional memory.

**EDID**
Extended Display Identification Data. A VESA standard that allows the display device to send identification and capabilities information to the Intel® Embedded Media and Graphics Driver. Intel® EMGD reads all EDID data, including resolution and timing data, from the display, thus negating the need for configuring DTD data for the device.

**EDID-less**
A display that does not have the capability to send identification and timing information to the driver and requires DTD information to be defined in the driver.

**eDP**
Embedded Display Port

**EFI**
Extensible Firmware Interface.

**eIA**
Embedded Intel® Architecture.

**EMI**
Electromagnetic Interference.

**ePOG**
Embedded Pre-OS Graphics Driver

**Extended Clone Mode**
A feature that allows you to have different sized displays in Clone mode.

**Framebuffer**
A region of physical memory used to store and render graphics to a display.

**GMA**
Intel® Graphics Media Accelerator. Refers to both the graphics hardware in Intel chipsets as well as the desktop/mobile driver. The GMA driver is not intended for use in embedded applications.

**GMS**
Graphics Mode Select (stolen memory).

**GOP**
Graphics Output Protocol

**HDCP**
High-bandwidth Digital-Content Protection. A specification that uses the DVI interface. HDCP encrypts the transmission of digital content between the video source (transmitter) and the digital display (receiver).

**HDMI**
High-Definition Multimedia Interface, an uncompressed, all-digital, audio/video interface.

**iDCT**
Inverse Discrete Cosine Transformation (hardware feature).

**LVDS**
Low Voltage Differential Signaling. Used with flat panel displays, such as a laptop computer display.

**Option ROM (OROM)**
Code that is integrated with the system BIOS and resides on a flash chip on the motherboard. The Intel Embedded Video BIOS is an example of an option ROM.

**OS**
Operating System.

**PCI**
Peripheral Component Interface.

**POST**
Power On Self Test.

**PWM**
Pulse Width Modulation.

**Reserved Memory**
A region of physical memory in a Windows Embedded Compact 7 system set aside for BIOS, VBIOS, and graphics driver operations. Reserved memory can be configured for use by the operating system and other applications when not in use by the BIOS.

---

### Table 1. Acronyms and Terminology (Sheet 2 of 3)

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contrast Ratio</td>
<td>Contrast ratio is the measure of the difference between light and dark on a display. If the contrast is increased, the difference between light and dark is increased (i.e., something white will be very bright and something black will be very dark). Brightness and Contrast Controls differ in function between CRTs and LCDs.</td>
</tr>
<tr>
<td>DC</td>
<td>Display Configuration.</td>
</tr>
<tr>
<td>DP</td>
<td>Display Port.</td>
</tr>
<tr>
<td>DTD</td>
<td>Detailed Timing Descriptor. A set of timing values used for EDID-less devices.</td>
</tr>
<tr>
<td>EBDA</td>
<td>Extended BIOS Data Area. An interface that allows the system BIOS and Option ROMs to request access to additional memory.</td>
</tr>
<tr>
<td>EDID</td>
<td>Extended Display Identification Data. A VESA standard that allows the display device to send identification and capabilities information to the Intel® Embedded Media and Graphics Driver. Intel® EMGD reads all EDID data, including resolution and timing data, from the display, thus negating the need for configuring DTD data for the device.</td>
</tr>
<tr>
<td>EDID-less</td>
<td>A display that does not have the capability to send identification and timing information to the driver and requires DTD information to be defined in the driver.</td>
</tr>
<tr>
<td>eDP</td>
<td>Embedded Display Port</td>
</tr>
<tr>
<td>EFI</td>
<td>Extensible Firmware Interface.</td>
</tr>
<tr>
<td>eIA</td>
<td>Embedded Intel® Architecture.</td>
</tr>
<tr>
<td>EMI</td>
<td>Electromagnetic Interference.</td>
</tr>
<tr>
<td>ePOG</td>
<td>Embedded Pre-OS Graphics Driver</td>
</tr>
<tr>
<td>Extended Clone Mode</td>
<td>A feature that allows you to have different sized displays in Clone mode.</td>
</tr>
<tr>
<td>Framebuffer</td>
<td>A region of physical memory used to store and render graphics to a display.</td>
</tr>
<tr>
<td>GMA</td>
<td>Intel® Graphics Media Accelerator. Refers to both the graphics hardware in Intel chipsets as well as the desktop/mobile driver. The GMA driver is not intended for use in embedded applications.</td>
</tr>
<tr>
<td>GMS</td>
<td>Graphics Mode Select (stolen memory).</td>
</tr>
<tr>
<td>GOP</td>
<td>Graphics Output Protocol</td>
</tr>
<tr>
<td>HDCP</td>
<td>High-bandwidth Digital-Content Protection. A specification that uses the DVI interface. HDCP encrypts the transmission of digital content between the video source (transmitter) and the digital display (receiver).</td>
</tr>
<tr>
<td>HDMI</td>
<td>High-Definition Multimedia Interface, an uncompressed, all-digital, audio/video interface.</td>
</tr>
<tr>
<td>iDCT</td>
<td>Inverse Discrete Cosine Transformation (hardware feature).</td>
</tr>
<tr>
<td>LVDS</td>
<td>Low Voltage Differential Signaling. Used with flat panel displays, such as a laptop computer display.</td>
</tr>
<tr>
<td>Option ROM (OROM)</td>
<td>Code that is integrated with the system BIOS and resides on a flash chip on the motherboard. The Intel Embedded Video BIOS is an example of an option ROM.</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System.</td>
</tr>
<tr>
<td>PCI</td>
<td>Peripheral Component Interface.</td>
</tr>
<tr>
<td>POST</td>
<td>Power On Self Test.</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation.</td>
</tr>
<tr>
<td>Reserved Memory</td>
<td>A region of physical memory in a Windows Embedded Compact 7 system set aside for BIOS, VBIOS, and graphics driver operations. Reserved memory can be configured for use by the operating system and other applications when not in use by the BIOS.</td>
</tr>
</tbody>
</table>
Monitors and scanners are based on the “additive” color system using RGB, starting with black and then adding Red, Green, and Blue to achieve color. Saturation is the colorfulness of an area judged in proportion to its brightness. Full saturation of RGB gives the perception of white, and images are created that radiate varying amounts of RGB, or varying saturation of RGB.

**SCS**  Software Compliance Statement.

**Single Display Configuration**  A type of display configuration that supports one and only one display device.

**SSC**  Spread Spectrum Clock.

**Stolen Memory**  A region of physical memory (RAM) set aside by the system BIOS for input and output operations. The amount of stolen memory is configurable. Stolen memory is not accessible to the operating system or applications.

**System BIOS**  The standard BIOS used for basic input and output operations on PCs.

**TOM**  Top Of Memory.

**TSR**  Terminate and Stay Resident. A program that is loaded and executes in RAM, but when it terminates, the program stays resident in memory and can be executed again immediately without being reloaded into memory.

**VBIOS**  Video Basic Input Output System. A component of system BIOS that drives graphics input and output.

**VESA**  Video Electronics Standards Organization.

**VGA**  Video Graphics Array. A graphics display standard developed by IBM® that uses analog signals rather than digital signals.

**VLD**  Variable Length Decoding.

**VMR**  Video Mixing Render.

**YUV**  Informal, but imprecise reference to the video image format, Y'CbCr. The Y' component is luma, a nonlinear video quality derived from RGB data denoted without color. The chroma components, Cb and Cr, correspond nonlinearly with U and V as differences between the blue and luma, and between the red and luma, respectively.
2.0 Architectural Overview

2.1 Introduction

The Intel® Embedded Media and Graphics Driver is composed of a runtime graphics driver and a video firmware component. (See Figure 1 below.) Both the driver and video firmware (i.e., VBIOS, GOP, and EPOG) control the processor to perform, display, and render operations. The video firmware is predominantly leveraged by System BIOS during system boot, but is also used at runtime by the driver to handle full-screen text mode on the operating system(s).

Figure 1. Intel® Embedded Media and Graphics Driver
2.1.1 Display Options

The following section describes the types of displays and configurations supported by Intel® EMGD.

2.1.1.1 Types of Displays

The table below lists the types of displays supported by the Intel® Embedded Media and Graphics Driver.

<table>
<thead>
<tr>
<th>Display Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRT Analog CRT. Also known as &quot;VGA&quot; typically using a 15 pin D-Sub connector.</td>
</tr>
<tr>
<td>HDMI High-Definition Multimedia Interface</td>
</tr>
<tr>
<td>DP Display Port</td>
</tr>
<tr>
<td>eDP embedded Display Port</td>
</tr>
</tbody>
</table>

2.1.1.2 Display Configuration

Intel® EMGD supports driving two displays simultaneously. Several configurations are supported, dependent on the operating system and processor. The various display configurations are described in Table 3.

<table>
<thead>
<tr>
<th>Display Configuration Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single</td>
<td>Normal desktop configuration, single monitor</td>
</tr>
<tr>
<td>Clone</td>
<td>Two displays, same content, different resolutions, independent timings</td>
</tr>
<tr>
<td>Extended</td>
<td>Two displays, different content, independent resolutions</td>
</tr>
</tbody>
</table>

The table below summarizes the display configurations supported by Intel chipsets.

<table>
<thead>
<tr>
<th>Supported Display Configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chipset</td>
</tr>
<tr>
<td>Intel® Atom™ Processor E3800</td>
</tr>
<tr>
<td>Product Family/ Intel® Celeron® Processors N2807/N2930/J1900</td>
</tr>
<tr>
<td>Operating System</td>
</tr>
<tr>
<td>Single, Clone, Extended</td>
</tr>
</tbody>
</table>
2.2 Features

The following sections describe major features that Intel® EMGD supports.

2.2.1 Processor(s) Supported

Table 5 below lists Intel® EMGD-supported processors.

Table 5. Processors Supported by Intel® EMGD v. 36.40.23 and Intel® EMGD v. 37.40.23 for Intel® Atom™ Processor E3800 Product Family/Intel® Celeron® Processors N2807/N2930/J1900 Linux* Release

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel® Atom™ Processor E3800 Product Family/Intel® Celeron® Processors N2807/N2930/J1900</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

All supported chipsets provide output on digital monitors. CRTs and TVs are supported through integrated display ports such as Display Port (DP), embedded Display Port (eDP), HDMI, interfaces, depending on hardware availability.

2.2.2 OS Support

Intel® EMGD supports the following operating systems and driver variants:

- Fedora* 32-bit
- Fedora* 64-bit

*Note:* For more information about version, X server, Mesa, DRI, and libva, refer to the driver package Release notes.

*Note:* Use the respective 32-bit Intel® EMGD driver/ GOP/ VBIOS/ EPOG for the 32-bit Fedora*; the 64-bit Intel® EMGD driver / GOP for 64-bit Fedora*. There are also two variants of CED Lite (32-bit and 64-bit), Binary_blob (32-bit and 64-bit), etc., that will pair with the respective 32-bit and 64-bit Fedora* OS.
3.0 Configuration and CED Lite

Intel® EMGD Configuration Editor Lite (CED Lite) is a browser-based application that allows you to create configurations for Linux* platforms. Configurations are associated with a specific processor or chipset and can be created for the following supported processor:

- Intel® Atom™ Processor E3800 Product Family/Intel® Celeron® Processor N2807/N2930/J1900

Note: CED Lite supports Intel® EMGD on Linux* and EPOG. Windows*, VBIOS, and EFI configurations are not supported in this application.

See also Creating a Configuration in CED Lite – Summary Steps and Environment Setup.

3.1 Environment Setup

The CED-lite application runs on both Linux* and Windows* environments.

**Launching CED Lite on Linux***

Perform the following steps to launch the CED-lite browser-based application on the Linux* GUI.

1. Install Python* 2.7 on your host system.
2. Set Firefox* as your default browser.
3. From the command line, change to the CED Lite home directory.
4. Execute the command `python launch.py`.

CED Lite opens in the system's default browser. To close the CED lite, press Ctrl+c at the terminal running it.

For generating Linux* configurations, no additional steps are required.

**Launching CED Lite on Windows***

Perform the following steps to launch CED Lite on a Windows* environment:

1. Install Python* 2.7 on your host system.
2. Set Firefox* as your default browser.
3. From the command line, change to the CED Lite home directory.
4. Execute the command `launch.py` at the command prompt.

CED Lite opens in the system's default browser.

Note: Make sure you execute only one instance of the launch.py script.

For configuring EPOG, copy the relevant EPOG tar package `IEMGD_HEAD_EPOG.tgz` into the folder `CED-Lite/pre_package/EPOG`. 
3.2 Creating a Configuration in CED Lite – Summary Steps

The following steps present a sample CED Lite configuration.

1. Create **New Configuration**.
   - Select the mode, chipset, ports, port drivers, DTDs, etc., for the configuration. For details, see Section 3.3, “Display Setup” on page 16.

2. (Optional) If you have custom panels and timings, you may want to create your own DTD; otherwise, you can use the standard DTDs provided by CED Lite. If needed, go to the General (Advance Settings) page Custom DTD tab.
   - In the General (Advance Settings) page Import DTD tab, Import a DTD that most closely aligns with your display parameters, click **Edit**, enter parameters, and then click **Add DTD** or **Save DTD**. Or, to create a DTD, see Section 3.10, “Creating a New Customized DTD” on page 23.

3. **Generate Installation**.
   After the required options are selected, click on Generate Intel® EMGD Configuration which generates the appropriate configuration files. For details, see Section 3.18, “Generating an Intel® EMGD Configuration” on page 32.

3.3 Display Setup

1. Select the Target OS for your configuration from the drop-down list.
2. Select the chipset for your configuration from the drop-down list.
3. Select **Single**, **Clone**, or **Extended** for the display. Note that EPOG works only in Single mode.
4. Select **HDMI**, **DP**, **eDP**, or **VGA** to indicate which is the primary display.

See also:
- “Entering Advanced Settings”
- “Entering Pipe Settings”
- “Setting Splash Screen Options”
- “CRTC Settings”
- “Importing a DTD”
- “Creating a New Customized DTD”
- “Setting HDMI, DP, eDP, and VGA Framebuffer Color Correction Attributes”
3.4 Entering Advanced Settings

The **General (Advanced Settings)** section allows you to enter Linux* options into the configuration. Table 6 describes each of these settings.

Table 6. Linux* OS Settings

<table>
<thead>
<tr>
<th>OS Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable Hardware Acceleration</td>
<td>Disable or enable hardware 2D acceleration. The default is to enable hardware acceleration. To disable acceleration, click the check box.</td>
</tr>
<tr>
<td>Disable Hardware Cursor</td>
<td>Disable the use of the hardware cursor. By default, the hardware cursor is enabled.</td>
</tr>
<tr>
<td>Enable Double Buffer (Shadow Framebuffer)</td>
<td>Enable double buffering on the framebuffer. By default, double buffering is disabled. To enable it, click the check box.</td>
</tr>
<tr>
<td>Disable XVideo Support</td>
<td>Disable XVideo support. In a dual independent head configuration, either the first display or the second display supports XVideo. Both displays cannot support XVideo simultaneously. The default for XVideo support is enabled.</td>
</tr>
<tr>
<td>Disable XVideo Blend</td>
<td>Disable XVideo support using the 3D blend manager. This provides XVideo support in configurations that cannot be supported with overlay. For example, this is supported on both displays in a dual independent head setup. It is also supported when the display is rotated or flipped. Color key is only supported if ShadowFB is enabled and the VideoKey is defined. The default is XVideoBlend support is enabled.</td>
</tr>
<tr>
<td>Enable Tear Frame Buffer</td>
<td>Makes the driver avoid a scheduled blit and gets content to the screen immediately when flip is not possible.</td>
</tr>
<tr>
<td>Disable Detection of Multiple DVOs</td>
<td>Disable the detection of multiple DVOs. By default, detection of multiple DVOs is enabled.</td>
</tr>
<tr>
<td>Disable Display Detection</td>
<td>Disable or enable display detection (port detection). By default, display detection is enabled.</td>
</tr>
<tr>
<td>XVideo Color Key</td>
<td>This sets the color key for XVideo and XVideoBlend. This value is either a 24-bit value or a 16-bit value, depending on the pixel depth of the screen. The color key is always enabled for XVideo, even when it is not defined. The color key is always disabled for XVideoBlend unless both this option is defined and the ShadowFB option is enabled. The default color key for XVideo is 0x0000ff00. For XVideoBlend, the color key is disabled by default.</td>
</tr>
</tbody>
</table>
3.5 Entering Pipe Settings

The **Pipe Settings** tab allows you to set the Plane Z-order for the layers shown on the display. Each pipe has three major planes: Display, first sprite, and second sprite. The Z-Order can be arranged using options described in Table 7.

**Table 7. Pipe Settings**

<table>
<thead>
<tr>
<th>Pipe A Settings/Pipe B Settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Plane</td>
<td>Click and drag the tiles to set the “Plane Z-Order”, which is the “order” in which each layer appears on screen. For example, if Sprite A is in front of Sprite B, then Sprite B would be obscured by any overlap with Sprite A.</td>
</tr>
<tr>
<td>Sprite A/ Sprite C</td>
<td></td>
</tr>
<tr>
<td>Sprite B/ Sprite D</td>
<td></td>
</tr>
<tr>
<td>Constant Alpha</td>
<td>Click and drag the slider control to set the transparency of the sprites for a specific pipe. The range is -127 to 127.</td>
</tr>
<tr>
<td>Enable Const Alpha</td>
<td>When checked, this option enables the <strong>Constant Alpha</strong> value set with the slider controls.</td>
</tr>
<tr>
<td>Enable Framebuffer Overlay Blending</td>
<td>When checked, this option enables overlay blending with the framebuffer on both display outputs and when display mode resolution is 32-bit XRGB.</td>
</tr>
</tbody>
</table>

See also:

- “Sprite Plane Settings”
- “Sprite Performance through VAAPI Stack with VAXV Support”
- “Flipping an OGL Surface onto Sprite Using Xprop”
3.6 Setting Splash Screen Options

Use the splash screen settings in Table 8 to customize how the selected splash screen image is displayed on the screen.

For EPOG, splash screen information must be defined before configuring the EPOG binary.

Table 8. Splash Screen Options

<table>
<thead>
<tr>
<th>Fastboot Settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quick Boot</td>
<td>The Quick Boot feature optimizes the speed that Intel® EMGD loads at the expense of compatibility and ease of use. Quick Boot disables non-critical features that affect the initialization time of the driver that are not needed for targeted embedded applications.</td>
</tr>
<tr>
<td></td>
<td>• 0x0=default;</td>
</tr>
<tr>
<td></td>
<td>• 0x1=Skip GTT;</td>
</tr>
<tr>
<td></td>
<td>• 0x2=Skips init for ports not in port order list;</td>
</tr>
<tr>
<td></td>
<td>• 0x3=sets both 0x1 &amp; 0x2</td>
</tr>
<tr>
<td>Enable Blt for Splash</td>
<td>This option enables the BLT function when a splash screen is enabled. If you are experiencing problems where the splash screen does not disappear after boot, make sure this option is enabled.</td>
</tr>
<tr>
<td>Splash Screen BG Color Red</td>
<td>Splash Screen BG Color Red must be between 0x0 and 0xFF.</td>
</tr>
<tr>
<td>Splash Screen BG Color Green</td>
<td>Splash Screen BG Color Green must be between 0x0 and 0xFF.</td>
</tr>
<tr>
<td>Splash Screen BG Color Blue</td>
<td>Splash Screen BG Color Blue must be between 0x0 and 0xFF.</td>
</tr>
<tr>
<td>Splash Screen X (upper left corner x coordinate)</td>
<td>The X location, in pixels, where the Firmware Splash Screen will be placed. This number is a signed number in 2’s complement. Positive numbers are offset from the left of the screen. Negative numbers are offset from the right of the screen.</td>
</tr>
<tr>
<td>Splash Screen Y (upper left corner y coordinate)</td>
<td>The Y location, in pixels, where the Firmware Splash Screen will be placed. This number is a signed number in 2’s complement. Positive numbers are offset from the top of the screen. Negative numbers are offset from the bottom of the screen.</td>
</tr>
<tr>
<td>Image File</td>
<td>Image file to be used as Splash Screen. The supported image formats are .bmp and .png.</td>
</tr>
</tbody>
</table>
3.7 Setting Splash Video Options

The Splash Video feature provides a mechanism to use a portion of the off-screen pre-allocated video memory ("Stolen Memory") as a video image that is displayed on an overlay to the framebuffer. The intention is that a video capture device external to Intel® EMGD will be configured to transfer a video stream to the configured location in video memory using DMA. The splash video remains in place until the Intel® EMGD is notified by an external application to disable the overlay.

Table 9. Splash Video Options

<table>
<thead>
<tr>
<th>Fastboot Settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Splash Video Offset</td>
<td>The offset, in bytes, from the base of video memory where the Splash Video will be placed. Care must be taken to ensure that this location is past the end of the on-screen framebuffer and that the full Splash Video image fits within the pre-allocated video memory. See “How to Select the Video_Offset” for more details.</td>
</tr>
<tr>
<td>Splash Video Pixel Format</td>
<td>The pixel format of the Splash Video image in memory. The available pixel formats are encoded values used within Intel® EMGD.</td>
</tr>
<tr>
<td>Splash Video Source Width</td>
<td>The width, in pixels, of the Splash Video image in memory.</td>
</tr>
<tr>
<td>Splash Video Source Height</td>
<td>The height, in pixels, of the Splash Video image in memory.</td>
</tr>
<tr>
<td>Splash Video Source Pitch</td>
<td>The pitch, in bytes, of the Splash Video image in memory. Pitch must be &gt;= bytes per pixel * source width.</td>
</tr>
<tr>
<td>Splash Video Destination Height</td>
<td>The height, in pixels, of the Splash Video window on the screen. This number must currently be the same as SrcHeight.</td>
</tr>
<tr>
<td>Splash Video Destination Width</td>
<td>The width of the screen. This number must currently be the same as SrcWidth.</td>
</tr>
<tr>
<td>Splash Video Destination X</td>
<td>The X location, in pixels, where the Splash Video will be placed. This number is a signed number in 2’s complement. Positive numbers are offset from the left of the screen. Negative numbers are offset from the right of the screen.</td>
</tr>
<tr>
<td>Splash Video Destination Y</td>
<td>The Y location, in pixels, where the Splash Video will be placed. This number is a signed number in 2’s complement. Positive numbers are offset from the top of the screen. Negative numbers are offset from the bottom of the screen.</td>
</tr>
</tbody>
</table>
### 3.7.1 Configuring Splash Video

The splash video feature can be used to display a video while the system is booting to the operating system. This section describes how to configure the required options.

#### Figure 2. Splash Video with 8 MB of Stolen Memory Example

The Video DMA area is where the video will be streamed. It is part of the stolen memory of our graphics device.

The external PCI Express* (PCIe*) device that is connected to the camera needs to know the exact DDR RAM physical address to stream or dump the video data at that memory location.

To calculate the Start DDR RAM physical address:

\[
\text{Start\_Phy\_Ram\_Addr} = \text{BGSM} + \text{Video\_Offset}
\]

where \text{BGSM} = \text{Base of Graphics Stolen Memory}

and \text{Video\_Offset} = \text{Offset where the video data is present. This is what you enter into the CED tool.}

To calculate BGSM, find the amount of physical RAM populated in the system: for example, 1 GB, and the stolen memory selected by the user in the system BIOS, for example, 8 MB.

\[
\text{BGSM} = 1\text{ GB} - 8\text{ MB} = 0x4000\text{ 0000} - 0x80\text{ 0000} = 0x3F80\text{ 0000}
\]
3.7.2 How to Select the Video_Offset

Determine the size of the maximum resolution of the framebuffer.

\[ \text{Size} = \text{framebuffer_height} \times \text{framebuffer_pitch} \]

where \( \text{framebuffer_pitch} = \text{framebuffer_width} \times \text{Bytes_per_Pixel} \) (page aligned)

For example, 1024x768 at 32-bit BPP:

\[ \text{Size} = 768 \times (1024 \times 4) = 3145728 = 0x300000 \]

For some usage models, the framebuffer pitch is set to 8192 bytes. In that case:

\[ \text{Size} = 768 \times 8192 = 6291456 = 0x600000 \]

The Video_Offset can start from \( 0x300000 \) or \( 0x600000 \) (if the pitch is 8192). See the notes below on the recommended values for the Video Offset.

Max Size of Splash Video = Size of Stolen Memory - Max Frame buffer size – Size of GTT – Size of Scratch Page (4 KB)

Notes:
1. For the Splash Video option the stolen memory MUST be a minimum of 8 MB. This is selected in the BIOS menu.
2. The recommended Video Offsets for the splash video are \( 0x600000 \) and \( 0x700000 \).
3. If the Size of the Video frame is more than 1 MB, choose \( 0x600000 \).

3.8 CRTC Settings

The CRTC settings are compile time values given by the user for the Display settings. These parameters are overridden by the module parameters after they are specified during runtime.

Table 10. Primary CRTC Settings

<table>
<thead>
<tr>
<th>Fastboot Settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Width</td>
<td>Number of pixels in a horizontal line on the display</td>
</tr>
<tr>
<td>Display Height</td>
<td>Number of pixels in a vertical line on the display</td>
</tr>
<tr>
<td>Display Refresh</td>
<td>Refresh rate of the display</td>
</tr>
</tbody>
</table>
| Show Advanced Framebuffer Settings | Shows the Framebuffer settings to the user. By default the Framebuffer settings are derived from the Display settings but the user can change the default settings:
  - Width - Number of pixels in a horizontal line in the framebuffer
  - Height - Number of vertical lines in the framebuffer
  - Depth - Number of bits used to represent color values in a single pixel
  - Color Quality - Number of bits used to represent a pixel in the framebuffer, 8, 16, 24, or 32. The default is 32. |
3.9 Importing a DTD

To import a DTD:
1. Browse to an XML file previously saved from CED Lite or Intel® EMGD Configuration Editor.
2. Click Add to include a new DTD in the Display Timing Descriptors list in the Port Settings page (see “Display Timing Descriptors (DTDs)” on page 28) that is based on the DTD file you selected above.

To edit a DTD:
1. Browse to an XML file previously saved from CED Lite or Intel® EMGD Configuration Editor.
2. Click Edit to upload the values from the DTD selected above into the controls present on Custom DTD tab.
3. Edit the values on the Custom DTD tab as needed (see Creating a New Customized DTD).
4. Choose one of the options below:
   — Click Add DTD to include the modified DTD in the Display Timing Descriptors list.
   OR
   — Click Save DTD to create a new DTD file.

3.10 Creating a New Customized DTD

CED Lite allows you to create Dynamic Timings Definitions (DTD) for EDID-less displays or displays for which you do not want to use the display's EDID settings. In either of those cases, you can create your own DTD using the steps below. Any DTDs you create will be available for all configurations. Alternatively, you can use one of the standard DTDs included in CED Lite.

1. On the Custom DTD tab, choose the DTD type from the drop-down list and then set parameters for your choice. Parameters for each type are described in the following sections.
2. Select the DTD Type that most closely aligns with your display parameters. Options are:
   • **EMGD Parameters** – The EMGD Parameters option allows the user to fill up the DTD using the standard laymen terms such as horizontal blank info, vertical blank width, etc.
   • **VESA Parameters** – The VESA Parameters allow the user to create a DTD from a VESA monitor timing standard.
   • **Hardware Parameters** – The Hardware Parameters are the parameters that are used by Intel® EMGD.
   • **Simple Parameters** – The Simple Parameters (CVT Standard) is a process for computing standard timing specifications. The method for developing Reduced Blanking timings is not included.
   • **Mode Lines** – The Mode Lines are a video timing spec used by X.Org. The X.Org timing setting for Mode Lines is “name” I A B C D E F G H.
   • For example: “640x480@8bpp” 25.175 640 672 728 816 480 489 501 526.
   • **EDID Block** – The EDID Block is the detailed timing section (18 bytes) of the basic 128-byte EDID data structure. The detailed timing section starts at 36h of the 128-byte EDID data structure. Enter the EDID block 1 byte at a time. Example: a0 0f 20 00 31 58 1c 20 d2 1a 14 00 f6 b8 00 00 00 18
Each of these DTD types may have additional parameters, which are described in Table 11.

Table 11. Intel® EMGD DTD Setting Options (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>DTD Parameter</th>
<th>Description</th>
<th>DTD Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTD Settings Flags</td>
<td>This section allows you to set flags for Interlace, Vertical Sync Polarity, Horizontal Sync Polarity, and Blank Sync Polarity. Each field in this section is described below. Note: These flags are Intel® EMGD-specific and do not correspond to VESA 3.0 flags.</td>
<td>EMGD, VESA, Hardware, Simple, Mode Lines</td>
</tr>
<tr>
<td>Interlaced Display</td>
<td>— Check for Interlaced — Cleared for Non-interlaced</td>
<td>EMGD, VESA, Hardware, Mode Lines</td>
</tr>
<tr>
<td>Vertical Sync Polarity</td>
<td>— Active Low (Default) — Active High</td>
<td>EMGD, VESA, Hardware, Mode Lines</td>
</tr>
<tr>
<td>Horizontal Sync Polarity</td>
<td>— Active Low (Default) — Active High</td>
<td>EMGD, VESA, Hardware, Simple, Mode Lines, EDID Block</td>
</tr>
<tr>
<td>Blank Sync Polarity</td>
<td>— Active Low (Default) — Active High</td>
<td>EMGD, VESA, Hardware, Simple, Mode Lines, EDID Block</td>
</tr>
<tr>
<td>Pixel Clock</td>
<td>Pixel clock value in KHz. Range 0-0x7fffffff.</td>
<td>EMGD, VESA, Hardware</td>
</tr>
<tr>
<td>Horizontal Sync Offset (Front Porch) in pixels</td>
<td>Specifies the amount of time after a line of the active video ends and the horizontal sync pulse starts (Horizontal Front Porch). Range 0-1024 [10 bits].</td>
<td>EMGD</td>
</tr>
<tr>
<td>Horizontal Sync Pulse Width (Sync Time) in pixels</td>
<td>Width of the Horizontal Sync Pulse (Sync Time) which synchronizes the display and returns the beam to the left side of the display. Range 0-8191[13 bits].</td>
<td>EMGD, VESA</td>
</tr>
<tr>
<td>Horizontal Blank Width (Blank Time) in pixels</td>
<td>This parameter indicates the amount of time it takes to move the beam from the right side of the display to the left side of the display (Blank Time). During this time, the beam is shut off, or blanked. Range 0-4096 [12 bits].</td>
<td>EMGD, VESA</td>
</tr>
<tr>
<td>Horizontal Active (Width) in pixels</td>
<td>Number of pixels displayed on a horizontal line (Width). Range 0-4096 [12 bits].</td>
<td>EMGD, VESA, Hardware, Simple</td>
</tr>
<tr>
<td>Horizontal Sync Start in pixels</td>
<td>This parameter specifies the start of the horizontal active time. Range 0-40957.</td>
<td>VESA, Hardware</td>
</tr>
<tr>
<td>Horizontal Sync End in pixels</td>
<td>This parameter specifies the end of the horizontal active time. Range 0-49148.</td>
<td>Hardware</td>
</tr>
<tr>
<td>Horizontal Blank Start in pixels</td>
<td>This parameter specifies the start of one line of the video and margin period. Range 0-32766.</td>
<td>VESA</td>
</tr>
<tr>
<td>Horizontal Blank End in pixels</td>
<td>This parameter specifies the end of one line of the video and margin period. Range 0-65533.</td>
<td>Hardware</td>
</tr>
<tr>
<td>Refresh in Hz</td>
<td>Also known as the Vertical Refresh, the rate the full display updates. Standard refresh rates are 50Hz, 60Hz, 75Hz, and 85Hz.</td>
<td>Simple</td>
</tr>
<tr>
<td>Vertical Sync Offset (Front Porch) in lines</td>
<td>Specifies the amount of time after last active line of video ends and vertical sync pulse starts (Vertical Front Porch). Range 0-64 [6 bits].</td>
<td>EMGD</td>
</tr>
<tr>
<td>Vertical Sync Pulse Width (Sync Time) in lines</td>
<td>Specifies the Width of the Vertical Sync Pulse which synchronizes the display on the vertical axis and returns the beam to the top, left side of the display. Range 0-64 [6 bits].</td>
<td>EMGD, VESA</td>
</tr>
<tr>
<td>Vertical Blank Width (Blank Time) in lines</td>
<td>The amount of time for the complete vertical blanking operation to complete. It indicates the time it takes to move the beam from the bottom right to the top, left side of the display (Blank Time). During this time, the beam is shut off, or blanked. Range 0-4096 [12 bits].</td>
<td>EMGD, VESA</td>
</tr>
</tbody>
</table>
Use **Add DTD** to include the new DTD in the Display Timing Descriptors list.

Use **Save DTD** to save the current configuration to a local file that can be imported later (see "Importing a DTD" on page 23).

See also “DTD Example Specifications”.

---
Table 11. **Intel® EMGD DTD Setting Options (Sheet 2 of 2)**

<table>
<thead>
<tr>
<th>DTD Parameter</th>
<th>Description</th>
<th>DTD Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertical Active (Height) in lines</td>
<td>The number of active lines displayed (Height). Range 0-4096 [12 bits].</td>
<td>EMGD, VESA, Hardware, Simple</td>
</tr>
<tr>
<td>Vertical Sync Start in lines</td>
<td>This parameter specifies the start of the vertical sync. Range 0-4157.</td>
<td>VESA, Hardware</td>
</tr>
<tr>
<td>Vertical Sync End in lines</td>
<td>This parameter specifies the end of the vertical sync. Range 0-4220.</td>
<td>Hardware</td>
</tr>
<tr>
<td>Vertical Blank Start in lines</td>
<td>This parameter specifies the start of display vertical blanking including margin period. Range 0-4094.</td>
<td>VESA</td>
</tr>
<tr>
<td>Vertical Blank End in lines</td>
<td>This parameter specifies the end of vertical blanking. Range 0-8189.</td>
<td>Hardware</td>
</tr>
<tr>
<td>Mode Lines</td>
<td>The Mode Lines are a video timing spec used by X.Org. The X.Org timing setting for Mode Lines is “name” I A B C D E F G H. For example: “640x480@8bpp” 25.175 640 672 728 816 480 489 501 526.</td>
<td>Mode Lines</td>
</tr>
<tr>
<td>EDID Block</td>
<td>The EDID Block is the detailed timing section (18 bytes) of the basic 128-byte EDID data structure. The detailed timing section starts at 36h of the 128-byte EDID data structure. Enter the EDID block 1 byte at a time. Example: a0 0f 20 00 31 58 1c 20 d2 1a 14 00 f6 b8 00 00 00 18</td>
<td>EDID Block</td>
</tr>
</tbody>
</table>
### 3.11 DTD Example Specifications

The following table shows example product specifications that can be used in the timing fields.

**Table 12. Timing Specification Example Values**

<table>
<thead>
<tr>
<th>Item</th>
<th>Symbol</th>
<th>Standard value</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clock</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frequency</td>
<td>1/ts</td>
<td>29.91</td>
<td>33.231</td>
<td>36.55</td>
<td>MHz</td>
<td></td>
</tr>
<tr>
<td>Period</td>
<td>ts</td>
<td>27.36</td>
<td>30.06</td>
<td>33.43</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>Hi-time</td>
<td>tsh</td>
<td>7</td>
<td>–</td>
<td>–</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>Low-time</td>
<td>tsl</td>
<td>7</td>
<td>–</td>
<td>–</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>DUTY ratio</td>
<td>th/tl</td>
<td>35</td>
<td>50</td>
<td>65</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td><strong>Data</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Setup time</td>
<td>tds</td>
<td>7</td>
<td>–</td>
<td>–</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>Hold time</td>
<td>tdh</td>
<td>4</td>
<td>–</td>
<td>–</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td><strong>H sync.</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Period</td>
<td>tlpl, tpdp</td>
<td>24.51</td>
<td>31.75</td>
<td>32.05</td>
<td>us</td>
<td></td>
</tr>
<tr>
<td>Pulse width</td>
<td>tlw</td>
<td>3</td>
<td>128</td>
<td>200</td>
<td>clk</td>
<td></td>
</tr>
<tr>
<td><strong>Enable</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Setup time</td>
<td>tdrs</td>
<td>7</td>
<td>–</td>
<td>–</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>Hold time</td>
<td>tdrh</td>
<td>4</td>
<td>–</td>
<td>–</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td><strong>V sync.</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Period</td>
<td>tfpl, tfpd</td>
<td>520</td>
<td>525</td>
<td>680</td>
<td>Line</td>
<td></td>
</tr>
<tr>
<td>Pulse width</td>
<td>tfw</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>Line</td>
<td></td>
</tr>
<tr>
<td><strong>V display</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Term</td>
<td>tvd</td>
<td>480</td>
<td>480</td>
<td>480</td>
<td>Line</td>
<td></td>
</tr>
<tr>
<td>Start</td>
<td>tfd</td>
<td>10</td>
<td>33</td>
<td>40</td>
<td>Line</td>
<td></td>
</tr>
<tr>
<td><strong>Phase difference</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H sync. ~ enable</td>
<td>tdrs</td>
<td>50</td>
<td>216</td>
<td>260</td>
<td>clk</td>
<td></td>
</tr>
<tr>
<td>H sync. ~ clock</td>
<td>tls</td>
<td>7</td>
<td>–</td>
<td>–</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

### 3.12 Sprite Plane Settings

The order in which each layer appears on screen is set in Section 3.5, “Entering Pipe Settings” on page 18.

Sprite Plane Settings allow the sprite plane to have color-correction settings that are different from the main framebuffer color-correction settings (see Section 3.15, “Setting HDMI, DP, eDP, and VGA Framebuffer Color Correction Attributes” on page 29). This feature lets you color-correct for red, green, and blue, plus it allows you to adjust brightness, contrast, and saturation. Table 13 describes each of these settings.

**Note:** The Sprite_A, Sprite-B, Sprite-C, or Sprite-D can be drag-and-dropped to the “gray” area section of a different “Plan Config” for different sprite configurations or to configure more than one config for the sprites.

**Table 13. Color Correction Values (applies to ALL color)**

<table>
<thead>
<tr>
<th>Option</th>
<th>Valid Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gamma:</td>
<td>0.6 to 6.0 (default value is 1)</td>
</tr>
<tr>
<td>Brightness:</td>
<td>0 to 200 (default value is 100)</td>
</tr>
<tr>
<td>Contrast:</td>
<td>0 to 200 (default value is 100)</td>
</tr>
<tr>
<td>Saturation:</td>
<td>0 to 200 (default value is 100)</td>
</tr>
</tbody>
</table>
Any sprite whose tile appears in the top Plane Config gray box is controlled by the options set in the Pipe Settings section. To customize a particular sprite, click the sprite tile from the Plane Config area at the top and drag it to an empty Plane Config gray box further down on the page. Drag the controls to adjust values for this sprite only.

See also:
- “Entering Pipe Settings”
- “Sprite Performance through VA-API Stack with VAXV Support”
- “Flipping an OGL Surface onto Sprite Using Xprop”

### 3.13 Configuring HDMI, DP, eDP and VGA Ports

From the **Port Settings** tab in the **HDMI Display**, **Display Port**, **eDP Display**, or **VGA Display (Advanced Settings)** sections you can specify whether to use EDID timings or customized DTD timings for the display connected to this specific port. In this section you can also specify attribute, I2C, and flat panel settings and create a new DTD that can be used with any configuration. **Table 14** describes each of these settings.

**Table 14. Port Settings (Sheet 1 of 2)**

<table>
<thead>
<tr>
<th>Port Configuration Field</th>
<th>Display Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port Name</td>
<td>HDMI, DP, eDP,</td>
<td>Enter a name for the port. This is a required field and the name must</td>
</tr>
<tr>
<td></td>
<td>VGA</td>
<td>be between 1 and 50 characters and may contain spaces.</td>
</tr>
<tr>
<td>Port Rotation</td>
<td>HDMI, DP, eDP,</td>
<td>This list allows you select a rotation for the display connected to this</td>
</tr>
<tr>
<td></td>
<td>VGA</td>
<td>port. You can choose between 0, 90, 180, and 270 degrees. The default is 0.</td>
</tr>
<tr>
<td>Disable standard timings</td>
<td>HDMI, DP, eDP,</td>
<td>This option disables the standard timings built into the graphics driver.</td>
</tr>
<tr>
<td>(EDID Options: Simple)</td>
<td>VGA</td>
<td></td>
</tr>
<tr>
<td>Disable EDID</td>
<td>HDMI, DP, eDP,</td>
<td>This option prevents the graphics driver from getting display timings</td>
</tr>
<tr>
<td>(timing from display)</td>
<td>VGA</td>
<td>from the display device.</td>
</tr>
<tr>
<td>(EDID Options: Simple)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
These options allow you to set EDID options for the display. Intel® EMGD supports three different types of EDID display modes:
- **Built-in display modes**: These modes are hard-coded in Intel® EMGD. These modes can be filtered based on the EDID block.
- **EDID Block**: These are Detailed Timing Descriptors read from an EDID display. An EDID display can contain DTD as well as other information about the display.
- **User-specified DTDs** (see Section 3.10, "Creating a New Customized DTD" on page 23).

If the display attached to this port contains EDID information, you can choose one or more of the following options from the **If EDID Device** section to determine which set of timings to use for the display connected to the port:
- Use driver built-in standard timings — If this box is checked, the standard timings built into Intel® EMGD are used.
- Use EDID block — If this box is checked, the EDID block is used.
- Use user-defined DTDs — If this box is checked, a user-defined DTD is used. You can select which DTD to use by checking the appropriate box in the Display Timings Descriptors (DTDs) section. If no DTDs are defined, you can create a custom DTD using instructions in Section 3.10, "Creating a New Customized DTD" on page 23.

If you select both **Use driver built-in standard timings** and **Use EDID block**, Intel® EMGD uses its built-in display timings and the timings provided by the display.

If the display attached to this port does not contain EDID information, you can choose one or both of the following options from the **If Not EDID Device** section:
- Use driver built-in standard timings — If this box is checked, the standard timings are used.
- Use user-defined DTDs — If this box is checked, a user defined DTD is used. You can select which DTD to use by checking the appropriate box in the Custom Display Timings Descriptors (DTDs) section. If no DTDs are defined, you can create a custom DTD using instructions in Section 3.10, "Creating a New Customized DTD" on page 23.

### Table 14. Port Settings (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Port Configuration Field</th>
<th>Display Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDID Options (Advanced)</td>
<td>HDMI, DP, eDP, VGA</td>
<td>These options allow you to set EDID options for the display. Intel® EMGD supports three different types of EDID display modes: <em>Built-in display modes</em>: These modes are hard-coded in Intel® EMGD. These modes can be filtered based on the EDID block. <em>EDID Block</em>: These are Detailed Timing Descriptors read from an EDID display. An EDID display can contain DTD as well as other information about the display. <em>User-specified DTDs</em> (see Section 3.10, &quot;Creating a New Customized DTD&quot; on page 23). If the display attached to this port contains EDID information, you can choose one or more of the following options from the <strong>If EDID Device</strong> section to determine which set of timings to use for the display connected to the port:  - Use driver built-in standard timings — If this box is checked, the standard timings built into Intel® EMGD are used.  - Use EDID block — If this box is checked, the EDID block is used.  - Use user-defined DTDs — If this box is checked, a user-defined DTD is used. You can select which DTD to use by checking the appropriate box in the Display Timings Descriptors (DTDs) section. If no DTDs are defined, you can create a custom DTD using instructions in Section 3.10, &quot;Creating a New Customized DTD&quot; on page 23.  If you select both <strong>Use driver built-in standard timings</strong> and <strong>Use EDID block</strong>, Intel® EMGD uses its built-in display timings and the timings provided by the display.  If the display attached to this port does not contain EDID information, you can choose one or both of the following options from the <strong>If Not EDID Device</strong> section:  - Use driver built-in standard timings — If this box is checked, the standard timings are used.  - Use user-defined DTDs — If this box is checked, a user defined DTD is used. You can select which DTD to use by checking the appropriate box in the Custom Display Timings Descriptors (DTDs) section. If no DTDs are defined, you can create a custom DTD using instructions in Section 3.10, &quot;Creating a New Customized DTD&quot; on page 23.</td>
</tr>
</tbody>
</table>
3.14 Configuring eDP or VGA Flat Panel Settings

From the Panel Settings tab in the eDP Display or VGA Display (Advanced Settings) sections you can specify settings for a flat panel display connected to a VGA or eDP port. Table 15 describes each option on this tab.

Table 15. Flat Panel Settings Options

<table>
<thead>
<tr>
<th>Flat Panel Settings</th>
<th>Applies to</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed Timing</td>
<td>eDP and VGA</td>
<td>This section indicates whether the attached display is a fixed timing display.</td>
</tr>
<tr>
<td>Centering and Upscaling</td>
<td>eDP and VGA</td>
<td>The Use Default check box lets you choose the default setting or either Upscaling or Force Centering. The default setting is upsampling.</td>
</tr>
<tr>
<td>Flat Panel Width</td>
<td>eDP</td>
<td>This option lets you configure a flat panel width with custom values: Min-640, Max-2560.</td>
</tr>
<tr>
<td>Flat Panel Height</td>
<td>eDP</td>
<td>This option lets you configure a flat panel height with custom values: Min-350, Max-1600.</td>
</tr>
<tr>
<td>Maintain Aspect Ratio</td>
<td>eDP</td>
<td>This option controls the scaled image to match the source image aspect ratio or full screen image.</td>
</tr>
<tr>
<td>Text Tuning</td>
<td>eDP</td>
<td>This option lets you set the text tuning using the following levels: DP_TEXT_TUNING_FUZZY, DP_TEXT_TUNING_CRISP, DP_TEXT_TUNING_MEDIAN.</td>
</tr>
<tr>
<td>Flat Panel Backlight Options</td>
<td>eDP</td>
<td>This section provides options for controlling the backlight of the flat panel display and specifying timing delays. The Backlight Control Methods list lets you choose either No Backlight or Port Driver to control the backlight. If you choose Port Driver, you can specify the timing delays in the Timing Delays section and the GPIO pin connections in the GPIO Pin Connections section. The default is No Backlight.</td>
</tr>
</tbody>
</table>

3.15 Setting HDMI, DP, eDP, and VGA Framebuffer Color Correction Attributes

Use the Color Correction tab to color-correct the display for red, green, and blue, and to adjust brightness and contrast. These settings are different from main framebuffer color-correction settings (see Section 3.12, "Sprite Plane Settings" on page 26). Table 16 describes each of these settings.

Table 16. Linux* Framebuffer Color Correction Values (applies to R, G, B color)

<table>
<thead>
<tr>
<th>Option</th>
<th>Valid Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gamma</td>
<td>0.6 to 6.0  (default value is 1)</td>
</tr>
<tr>
<td>Brightness</td>
<td>-127 to 127 (default value is 0)</td>
</tr>
<tr>
<td>Contrast</td>
<td>-127 to 127 (default value is 0)</td>
</tr>
</tbody>
</table>

Drag the controls to adjust values, which are shown above the slider controls.
3.16 Configuring HDMI, DP, eDP, or VGA Attribute Settings

When you click the Attributes Settings tab in the HDMI Display, Display Port, eDP Display, or VGA Display (Advanced Settings) sections, a page of attributes for the selected encoder device is displayed. The actual page that appears depends upon the encoder device selected and only the attributes that apply to the selected encoder appear.

To add a custom attribute:

1. Click Add Custom Attribute. A new attribute line opens in the Custom Attribute section.
2. Click the check box in front of the new attribute and then type contents for the ID, Value, and Name/Description fields.
3. When you are finished adding custom attributes, save the settings (see Section 3.19, “Options” on page 32).

For a full description of all attributes for all supported encoders, refer to Section 3.17, “Port Driver Attributes” on page 30.

3.17 Port Driver Attributes

3.17.1 Standard Port Driver Attributes

Intel® Embedded Media and Graphics Driver incorporates the port devices capability into the driver. Table 17 lists the attributes available to port drivers. Some of these standard attributes can be customized for specific port drivers and are detailed in the following topic:

- “HDMI Port Driver Attributes” on page 31

Note: Not all standard attributes are supported by all port drivers. See the topic mentioned above for details on the specific attributes supported by each port driver. Flat panel settings are specified via the FPINFO options of the configuration.

Table 17. Standard Port Driver Attributes (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute ID Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PANEL FIT</td>
<td>18</td>
<td>Panel fitting. Yes or no.</td>
</tr>
<tr>
<td>PWR T1</td>
<td>21</td>
<td>eDP T3 delay from LCDVDD to HDP high.</td>
</tr>
<tr>
<td>PWR T2</td>
<td>22</td>
<td>eDP T7 delay from valid video data from source to video on display.</td>
</tr>
<tr>
<td>PWR T3</td>
<td>23</td>
<td>eDP T9 delay from backlight disable to end of valid video data.</td>
</tr>
<tr>
<td>PWR T4</td>
<td>24</td>
<td>eDP T10 delay from end of valid video data from source to power off.</td>
</tr>
<tr>
<td>PWR T5</td>
<td>25</td>
<td>eDP T12 power off time.</td>
</tr>
<tr>
<td>FRAMEBUFFER GAMMA</td>
<td>35</td>
<td>Framebuffer gamma correction.</td>
</tr>
<tr>
<td>FRAMEBUFFER BRIGHTNESS</td>
<td>36</td>
<td>Framebuffer brightness.</td>
</tr>
<tr>
<td>FRAMEBUFFER CONTRAST</td>
<td>37</td>
<td>Framebuffer contrast.</td>
</tr>
<tr>
<td>TEXT ENHANCEMENT</td>
<td>58</td>
<td>Controls text tuning.</td>
</tr>
<tr>
<td>MAINTAIN ASPECT RATIO</td>
<td>59</td>
<td>This controls scaled image to match source image aspect ratio or full screen image.</td>
</tr>
<tr>
<td>FIXED TIMING</td>
<td>60</td>
<td>This indicates whether the attached display is a fixed timing display.</td>
</tr>
</tbody>
</table>
3.17.2 HDMI Port Driver Attributes

3.17.2.1 Audio

The Intel® EMGD package does not include an HDMI audio driver, so you must obtain and install the driver yourself. The HDMI audio driver needs to support Intel® High Definition Audio (Intel® HD Audio) to be compatible with Intel® EMGD.

3.17.2.2 Internal HDMI

Only one HDMI port has audio at any one time. The first port in the port order has audio while the second port would have only display without audio.

Only one HDMI port has HDCP at any one time. The first port to receive a request for HDCP has HDCP enabled only in that port.

3.17.3 Default Search Order

Table 18. Default Search Order

<table>
<thead>
<tr>
<th>Chipset</th>
<th>Default Search Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processor N2807/N2930/J1900 platform</td>
<td>CRT, HDMIC, HDMIB, DPB, DPC</td>
</tr>
</tbody>
</table>

3.17.4 Default GPIO Pin Pair Assignments

Table 19. Default GPIO Pin Pair Assignments

<table>
<thead>
<tr>
<th>Chipset</th>
<th>Default GPIO Pin Pair for EDID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processor N2807/N2930/J1900</td>
<td>HDMI-B 4</td>
</tr>
</tbody>
</table>

Table 17. Standard Port Driver Attributes (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute ID Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PD_ATTR_ID_PWM_INTENSITY</td>
<td>70</td>
<td>This option is used to set eDP inverter frequency for backlight PWM in Hz. Valid value for the frequency is 100Hz to 20,000 Hz (20kHz).</td>
</tr>
<tr>
<td>DPCD MAX</td>
<td>74</td>
<td>DisplayPort maximum DPCD (DisplayPort Configuration Data) rate.</td>
</tr>
<tr>
<td>S3D MODE SET</td>
<td>75</td>
<td>This attribute is being exported as a connector property and is used to specify the stereo 3D mode for mode setting.</td>
</tr>
<tr>
<td>S3D MODE EXPOSE</td>
<td>76</td>
<td>This attribute is being exported as a connector property and is used to exposed modes that support stereo 3D.</td>
</tr>
<tr>
<td>PWR METHOD</td>
<td>78</td>
<td>eDP power method. Yes or no.</td>
</tr>
<tr>
<td>USER BPC</td>
<td>79</td>
<td>DisplayPort user input for panel bits per color setting. Available inputs are 6, 8 and 10 bpc.</td>
</tr>
<tr>
<td>DPCD BPC</td>
<td>80</td>
<td>DisplayPort information built-in inside the panel firmware for bits per color setting.</td>
</tr>
</tbody>
</table>
3.18 Generating an Intel® EMGD Configuration

After selecting the desired configuration options, generate the configuration file by clicking Generate Intel EMGD Configuration.

To generate a configuration for Linux*:
All the generated config files (user_config.txt, image_data.txt, xorg.txt) can be found inside the folder installation/Linux in the CED Lite home folder.

To generate a configuration for EPOG:
The configured EPOG binary output_libepog.a can be found inside the folder installation/EPOG/Compiled_package.

3.19 Options

The Options window contains information about CED Lite, including the target path for the configuration and the build version number.

From this window you can also save or load settings to/from your browser or to/from a file. Additionally, you can set the save and load settings and select the Save options.

Using the Save and Load Settings the user can select whether to Save/Load a settings file from the browser or from an external location.

3.20 Reset

To reset all configurations to defaults, click Reset on the main window.

3.21 New xorg options in EMGD SNA:

Note: All the configuration are optional.

3.21.1 Plane Reservation xorg configuration:

These options can be used to reserve a sprite plane for use by applications using DRI or Xv API. The options are optional.

If any of the options is not set, the default reservation value of ANY is used for the plane.

In the following example:

— Plane A is reserved for use by DRI API.
— Plane B is reserved for use by Any API.
— Plane C is reserved for use by Xv API.
— Invalid reservation value. Plane D is reserved for use by ANY API.

Example xorg.conf options:

Option "PlaneAReservation" "DRI"
Option "PlaneBReservation" "ANY"
Option "PlaneCReservation" "Xv"
Option "PlaneDReservation" "invalid_keyword"
3.21.2 XvAdapter xorg configuration

This option allows the user to select which adapters among Sprite adapter, Textured adapter, and Overlay adapter should be setup and made available to Xorg applications.

In the following example Texture and Sprite Adapters are initialized, with the Textured ordered first.

Example xorg.conf option:

```
Option XvAdapters "Textured, Sprite"
```

Note: Plane z-order configuration (refer Section 3.5) and plane Colour correction (refer Section 3.15) is configurable using CED-lite.
4.0 Firmware: GOP, EPOG, and VBIOS

4.1 Overview

The Intel® Embedded Video firmware (GOP, EPOG, and VBIOS) incorporates many of the features and capabilities of the Intel® Embedded Media and Graphics Driver. Intel® EMGD supports three variants of video firmware for different working environments, where one pairs GOP with SBIOS (UEFI), EPOG with OTM and VBIOS with Legacy SBIOS. The video firmware includes support for the following processor:

- Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processor N2807/N2930/J1900

EPOG uses CED Lite for configuration, whereas GOP and VBIOS use the BMP tool. For more information about EPOG, GOP, and VBIOS configuration, refer to the respective BMP document (for GOP and VBIOS stated in “Related Documents” on page 8) and “Configuration and CED Lite” on page 15 (for EPOG configuration). This User Guide mainly discusses configuration of the EPOG driver.

4.2 System Requirements for Video Firmware

The target system must contain the following:

- Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processor N2807/N2930/J1900 platform
- A minimum of 64 MB of RAM.

See also:

“Embedded Pre-OS Graphics (EPOG) Driver”

4.3 Embedded Pre-OS Graphics (EPOG) Driver

4.3.1 EPOG Overview

The EPOG driver is responsible for initializing the graphics chipset and displaying a splash screen. PCI enumeration and resource allocation for the IGD device must be complete prior to the invocation of the EPOG driver. The EPOG driver is built using the GNU Compiler Collection (GCC) 4.1.2.

*Note:* EPOG is currently supported only on 32-bit systems.
4.3.2 System Requirements
The minimum system requirements for configuring the EPOG with CED Lite:
- Intel® Atom™ Processor E3800 Product Family/Intel® Celeron® Processor N2807/N2930/J1900 platform with Intel® EMGD installed with CED Lite
- Firefox* web browser version 15 (recommended) or above
- Python* version 2.7 (recommended) or above

4.3.3 Configuring EPOG through CED Lite
To properly configure an EPOG image, you must do a minimum of the following settings:
1. Launch CED Lite.
2. Select the Target OS and configure the Display output type.
3. Configure the General Display settings and choose a splash screen.
4. Configure the specific Display output settings.
5. Generate the EPOG binary image using Generate the Intel® EMGD Configuration.

4.3.3.1 EPOG Configuration Examples
Use the following examples and steps to create a functional EPOG configuration.
4.3.3.1.1 Launch CED Lite

1. Download the latest CED Lite tgz package.
2. Extract the contents of the package to a working directory:
   `tar -xvzf <Package name>.tgz`
3. Go to the package Directory, which is `<EMGD package directory>` in this example.
4. Copy the file `<EMGD package directory>/EPOG/IEMGD_HEAD_EPOG.tgz` to the directory `<EMGD package directory>/CED-Lite/pre_package/EPOG`.
5. Launch CED Lite by double-clicking the `<EMGD package directory>/CED-Lite/launch.py`, with a system command line interface (e.g., in Windows launch CED Lite from command prompt by executing `launch.py`). CED Lite should open in the default web browser on the user's system.

*Note:* Intel recommends changing the default web browser to Firefox* to ensure full compatibility with CED Lite.

*Note:* You may encounter the following error while trying to launch CED Lite by clicking on `index.html` in the CED Lite directory. Make sure to launch it using `launch.py` as mentioned previously.

4.3.3.1.2 Select the Target OS and Configure the Display Output Type

1. After the CED Lite page is opened in a web browser, select the Target OS as EPOG.
2. Select the display type to configure EPOG using the Primary Display drop-down menu, and the connector type using the Connector options (either B or C, except when a VGA display is selected).
3. Expand the General (Advanced Settings) pane.
4. Select the Splash Screen tab and then enter the background color (BG) for each of the red, green, and blue (RGB) color components (0x00 to 0xFF/0xff).

   **Note:** For more details on the splash screen settings, refer to Section 3.6, “Setting Splash Screen Options” on page 19.

5. Select the x and y displacement for the splash screen offset (in pixels) from the upper left corner of the selected display type.
6. Use **Browse** to select a BMP format image, which must not exceed 500KB in size.

*Note:* EPOG supports only BMP 24 bits per pixel images with no palette information for the splash screen.
4.3.3.1.3 Configure Specific Display Output Settings

1. Select the Display (Advanced Settings) tab and then enter a name for the display/port type.
2. To modify the port rotation, select the number of degrees, in increments of 90, counter-clockwise.
   To rotate the display, select a value from the Port Rotation drop-down menu.
3. For EDID Options:
   a. Select Advanced from the drop-down list.
   b. Select a display timing descriptor (DTD) from the Display Timing Descriptors (DTDs) list.
   c. Clear the check box Use driver built-in standard timings.
   d. Under the section If Not EDID Device, select the check box Use user-defined DTDs.
4.3.3.1.4 Generate the EPOG Binary Image

1. To create the custom EPOG image, click **Generate Intel® EMGD Configuration** located at the top or bottom of the page. After the message “EPOG_Configuration.py saved successfully” appears, click **OK** to continue.

2. Navigate to the user-configured EPOG file at .../CED-Lite/installation/EPOG/Compiled package and then copy output_libepog.a for integration with further system firmware packaging.

3. After the binary output is generated, merge it into the respective boot loader.
5.0 Installing Linux* OS Drivers

Important: This assumes you have already fully built the Fedora* 18 Kernel 3.8 on Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processor N2807/N2930/J1900 platform (refer to Appendix A for more details about bringing up the Fedora* 18 Kernel 3.8 on Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processor N2807/N2930/J1900 platform).

5.1 Overview

This topic describes the installation and configuration of the Intel® Embedded Media and Graphics Driver for (Intel® EMGD) Linux*. The instructions provided here are applicable for 32-bit and 64-bit operating systems, unless otherwise specifically mentioned. Refer to the product release note for more details on the driver capabilities.

The Intel Linux* driver is for use with the Intel® Integrated Embedded Architecture graphics hardware:

This section provides the following information:

- Pre-requisite and supported hardware requirements/software for Intel® EMGD (see Appendix A, "Fedora* 18 Bring-up with Kernel 3.8 on Intel® E3800/N2807/N2930/J1900")
- Installation of Intel® EMGD on a Fedora* platform (see Appendix A, "Fedora* 18 Bring-up with Kernel 3.8 on Intel® E3800/N2807/N2930/J1900")
- Configuring and installing the user space libraries on Fedora
- Configuring Intel® EMGD on Linux* OS using CED Lite and generating binary-blob
- Xwindows and GUI
- Runtime configurations for Intel® EMGD
- Xrandr and sysfs
- Sprite Performance through VAAPI Stack with VAXV Support
- Flipping an OGL Surface onto Sprite Using Xprop
- Enabling HDMI audio
- Enabling DisplayPort Audio
- Enabling RC6 feature
- EMGD S3D mode
- Enabling Hot-Plug
See also:

- “Pre-requisites, Supported Hardware and Software Requirements for Intel® EMGD”
- “Supported Hardware”
- “Installation of Intel® EMGD on a Fedora* Platform”
- “Configuring and Installing the User Space Libraries on Fedora*”
- “Configuring the Intel® EMGD on Linux* OS Using CED Lite and Generating Binary Blob”
- “The Linux* OS Configuration File”
- “Runtime Configuration GUI”
- “Xrandr”
- “Intel® EMGD Custom Control Using the sysfs Interface”

5.2 Pre-requisites, Supported Hardware and Software Requirements for Intel® EMGD

The pre-requisites for installing and configuring Intel® EMGD for Linux* are as follows:

- Freshly installed Fedora* 18 GNOME-based Desktop Edition with Kernel 3.8 and I/O patches applied. Refer to Appendix A for information about how to install Fedora* along with Kernel 3.8 and patches.
- Latest SBIOS and video firmware installed on the system.
- Intel® EMGD Package
- System administration privileges
- Internet connection (For downloading the required patches, etc.)

5.2.1 Supported Hardware

The target system must contain the Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processor N2807/N2930/J1900 chipset/processor.

5.3 Installation of Intel® EMGD on a Fedora* Platform

To install Intel® EMGD on a Fedora platform:

1. Boot up the system and start a terminal, unless you are already in console mode. Switch to root:
   
   $su

2. Enter the root password.

3. Untar the Intel® EMGD release package in a convenient location using the following command:

   #tar -xvzf <EMGD release package.tgz>

4. Change your directory to the Linux sub folder and extract the IEMGD_HEAD_Linux.tgz file using the following commands.

   #cd /<Path to the emgd_package_directory>/Linux/
   #tar -xvzf IEMGD_HEAD_Linux.tgz
5. Change your directory to the common/drm sub-directory, untar emgd.tgz, and build the Intel®
EMGD Direct Rendering Manager (DRM) module.

```
#tar -xvzf emgd_drm.tgz
#make
#make install
#modprobe emgd
```

6. Another option is to install the EMGD DRM by using the RPM package as follows:
   a. Change your directory to the Linux sub folder and extract the IEMGD_HEAD_Linux.tgz file
      using the following commands:
         ```
         #cd /<Path to the emgd_package_directory>/Linux/
         #tar -xvzf IEMGD_HEAD_Linux.tgz
         ```
   b. Then, enter Fedora18 sub directory.
      ```
      cd /<Path to the emgd_package_directory>/Linux/Fedora18/
      ```
   c. Use the following command to install the rpm.
      ```
      #rpm --nodeps emgd-drm-2.0-<build number>.i686.rpm
      ```

7. Change the directory to Linux/Fedora18 and then install EMGD DDX using the following
   commands:
   ```
   #rpm --nodeps emgd-sna-2.0-<build number>.i686.rpm
   ```
   (Run the command #rpm --nodeps emgd-dmx-2.0-<build number>.i686.rpm for
   SNA-based (Sandybridge's New Acceleration) DDX)

8. Copy the xorg.txt file generated by CED to your /etc/X11 directory and save it as xorg.conf
   as follows:
   ```
   #cp /<path_for_xorg.txt>/ /etc/x11/xorg.conf
   ```
   Xorg.conf will be further discussed in Section 5.5, "Configuring the Intel® EMGD on Linux* OS
   Using CED Lite and Generating Binary Blob" on page 56.

9. Reboot

10. Confirm that you are using the updated Intel® EMGD drivers by opening the terminal and using
    the command:
    ```
    #rpm -qa | grep emgd
    ```
    To uninstall Intel® EMGD, use the following command:
    ```
    #rpm -e emgd-ddx emgd-drm
    ```

**Note:** Update the X server version to use SNA driver. Run the following command to get the latest
version of the X server (ver 1.13.3-3.fc18)
```
#yum install xorg-x11-server-Xorg
```
Or you can set the below lines in xorg.conf to startX since the SNA does not support the older
version 1.13.0
```
Section "ServerFlags"
Option "IgnoreABI" "1"
EndSection
5.4 Configuring and Installing the User Space Libraries on Fedora*

5.4.1 Installing Dependencies for User Space Libraries

Use the following commands to install the dependencies for the user space libraries.

```
# yum groupinstall "Development Tools"
# yum install flex bison imake libtool xorg-x11-proto-devel libdrm-devel gcc-c++
xorg-x11-server-devel libXi-devel libXmu-devel libXdamage-devel git expat-devel
llvm-devel libudev-devel
```

5.4.2 Installing the Direct Rendering Manager (DRM) Library

The `libdrm` file provides core library routines for the X Window* system to directly interface with video hardware using the Linux* kernel’s Direct Rendering Manager (DRM). Use the following instructions to install the `libdrm` on the Intel® Atom™ Processor E3800 Product Family/Intel® Celeron® Processor N2807/N2930/J1900 platform.

**Note:** Uninstall any previously installed drm libraries before installing libdrm-2.4.52. The uninstall can be done by using the command `rpm -e <package name>`.

1. Download the package `libdrm-2.4.52` from the following source:
   [http://cgit.freedesktop.org/mesa/drm/snapshot/libdrm-2.4.52.tar.gz](http://cgit.freedesktop.org/mesa/drm/snapshot/libdrm-2.4.52.tar.gz)

2. The `libdrm` patch is available in the EMGD package. Apply the patch with the following command:
   ```
   # patch --p1 <Linux/patches/common
   libDRM_2_4_52_VLV_Support.patch
   ```

3. Run the following commands to configure and install the `libDRM` package:
   ```
   # ./autogen.sh --prefix=/usr
   --disable-radeon --disable-nouveau --disable-vmwgfx
   # make
   # make install
   ```

The section “Installing 3-D Components” describes methods of configuring the Intel® EMGD for the Linux* operating system.

See also The Linux* OS Configuration File for details on the file structure.
5.4.3 Installing 3-D Components

5.4.3.1 Installing Mesa

Mesa is an open-source implementation of the OpenGL specification; a system that renders interactive 3-D graphics. Follow these steps to install Mesa on the Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processor N2807/N2930/J1900 platform:

1. Download the Mesa 10.0.1 package from the following source:
   http://cgit.freedesktop.org/mesa/mesa/snapshot/mesa-10.0.1.tar.gz
2. The Mesa patch is available in the EMGD package with the following command:
   #patch -p1 < Linux/patches/common/MESA_i965/<relevant MESA version patch file>.patch

Note: The command above is an example only. The file name for the patch may change for future releases.
3. Run the following commands to configure and install the Mesa 10.0.1 package:
   #./autogen.sh --prefix=/usr --enable-opengl --enable-egl --enable-gles1
   --enable-gles2 --with-egl-platforms=x11,drm --enable-shared-glapi
   --with-dri-drivers=i915,i965 --with-gallium-drivers= --disable-gallium-egl
   --disable-gallium-llvm
   #make
   #make install

5.4.3.2 EMGD OpenGL Support

Intel® EMGD supports OpenGL* for the following Intel chipsets:
- Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processor N2807/N2930/J1900

The OpenGL implementation for Intel® EMGD consists of three components:
- Mesa DRI module (i965_dri.so) or UFO DRI module (iu3d_dri.so)
- Kernel DRM module (emgd.ko)
- Device Dependent X (DDX) driver (emgd_drv.so)

Installing the Intel® EMGD OpenGL driver provides a fully hardware-accelerated implementation of the OpenGL library to applications. This implementation makes use of a direct rendering technology, which allows the client to directly write to DMA buffers that are used by the graphics hardware.

Due to the use of direct rendering technology, system designers must take special care to ensure that only trusted clients are allowed to use the OpenGL library. A malicious application could otherwise use direct rendering to destabilize the graphics hardware or, in theory, elevate their permissions on the system.

A system designer can control access to the direct rendering functionality by limiting the access to the DRI device file.
5.4.3.2.1 Installing the Intel® Unified 3D (UFO) Driver

The Intel® Unified 3D (UFO) is a closed source implementation of 3D components. Intel recommends using UFO for better performance. Follow the steps below to install the UFO:

1. Download the Intel® EMGD official full package that contains the UFO driver inside.
2. Enter the terminal and extract the Intel® EMGD package:
   
   ```
   #tar -xvzf <EMGD_Package>.tgz
   ```
3. Change directories to the UFO folder and then untar the UFO package:
   
   ```
   #tar -xvzf IEMGD_HEAD_UFO.tgz
   ```
   
   A library file named `iu3d_dri.so` is extracted.
4. Copy `iu3d_dri.so` to `/usr/lib/dri` directory:
   
   ```
   #cp iu3d_dri.so /usr/lib/dri
   ```
5. Backup the old `i965_dri.so` file:
   
   ```
   #mv i965_dri.so i965_dri_mesa.so
   ```
6. Create a soft link:
   
   ```
   #ln -sfv iu3d_dri.so i965_dri.so
   ```
7. Start X server and verify that UFO is installed:
   
   ```
   #/usr/bin/startx
   #glxinfo | grep "renderer"
   ```

**Note:** To remove Intel® Unified 3D, go to `/usr/lib/dri` and remove the soft link `i965_dri.so`, then replace the old `i965_dri.so` that you had backed up (in this case, `i965_dri_mesa.so`) and rename it to `i965_dri.so`. Next, execute `startx`.

5.4.3.3 Installing OpenGL Utility Library (GLU)

The OpenGL Utility Library (GLU) is a graphics library for OpenGL. Follow the procedure below to install GLU on Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processor N2807/N2930/J1900 platform.

1. Download the package named `glu-9.0` from the following source:
   
   ```
   http://cgit.freedesktop.org/mesa/glu/snapshot/glu-9.0.0.tar.gz
   ```
2. Run the following commands to configure and install the `glu-9.0` package.

   ```
   #./autogen.sh --prefix=/usr
   #make
   #make install
   ```

5.4.4 Configuring Accelerated Video Encode and Decode for Intel® EMGD

5.4.4.1 Hardware Video Acceleration (VA) Overview

Hardware Video Acceleration is the use of a specialized video engine to decode video streams (such as MPEG-2, H.264, and VC-1) to free up the processor from having to do all of the decoding.

1. The video player, such as the Intel® EMGD-validated GStreamer-VAAPI, reads a video file with the type specified through the GStreamer parameters command.
2. Based on type, the proper codec shared library object is loaded.
3. The codec loads the VA library shared library object.
4. The VA library loads the `emgd_drv_video.so` shared library object.
5. The `emgd_drv_video.so` communicates over the X wire protocol with the Intel® EMGD X driver to send encoded video to the hardware for decoding.

See also:
- “Installing VA Library”
- “Installing and Configuring Video-based Preparation - Intel® Driver”
- “Installing and Configuring GStreamer VA-API”
- “Video Encode with USB Camera”

5.4.4.2 Installing VA Library

The `libva` is an implementation of Video Acceleration API (VA-API) for Linux®. Follow the procedure below to install `libva` on the Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processor N2807/N2930/J1900 platform.

1. Choose one of the download options below:
   a. Download the package, `libva 1.2.1`, as release tar.gz or git clone from the source below. Currently, the 1.2.1 version is supported. http://cgit.freedesktop.org/libva/snapshot/libva-1.2.1.tar.gz
   b. Untar the tar ball:
      
      ```
      #tar zxf <filename>.tar.gz
      ```

      OR
   a. Use `git clone` to check out the `libva` as follows:
      
      ```
      #git clone git://anongit.freedesktop.org/libva
      #cd libva
      #git checkout 88ed1ebe960b1c4a7970e12f8df1ed7d7031352a
      #git log (Shows commit logs)
      ```

2. Move to the libva directory:

   ```
   #cd <libva-directory>
   ```

   You may have to set the prefix to `/usr` if you want the library installed to `/usr`.

3. Run the following commands to install and configure the `libva 1.2.1` package.
   ```
   #./autogen.sh --prefix=/usr
   #make
   #make install
   ```

See also:
- “Hardware Video Acceleration (VA) Overview”
- “Installing and Configuring Video-based Preparation - Intel® Driver”
- “Installing and Configuring GStreamer VA-API”
- “Video Encode with USB Camera”
5.4.4.3 Installing and Configuring Video-based Preparation - Intel® Driver

The Intel-VA Driver supports the HW video decode on Intel® Atom™ Processor E3800 Product Family/Intel® Celeron® Processor N2807/N2930/J1900 platform. Perform the following steps to install Intel-VA driver.

1. Choose one of the download options below:
   a. Download the package, intel-driver-1.2.2 as release tar.gz or git clone from the source below. Currently 1.2.2 version is supported.
      http://cgit.freedesktop.org/vaapi/intel-driver/snapshot/intel-driver-1.2.2.tar.gz
   b. Untar the tar ball:
      ```
      #tar zxf <filename>.tar.gz
      ```

2. Move to the driver directory:
   ```
   #cd <intel_driver>
   ```

3. Intel’s driver patch is available in the EMGD package. Patch using the following commands:
   ```
   #patch -p1 <Linux/patches/common/VA_Driver_i965/
   Intel_VA_Driver_1_2_2_VLV_Support.patch
   ```
   
   Note: The command above is an example only. The file name for the patch may change for future releases.

4. Build and install the driver:
   ```
   #./autogen.sh --prefix=/usr
   #make
   #make install
   ```
   
   Note: To make the Intel video driver work for Intel® EMGD, i965_drv_video.so must be linked as emgd_drv_video.so by using the commands below:

5. Change the directory and create a link:
   ```
   #cd /usr/lib/dri/
   #ln -sfv i965_drv_video.so emgd_drv_video.so
   ```

See also:
- "Hardware Video Acceleration (VA) Overview"
- "Installing VA Library"
- "Installing and Configuring GStreamer VAAPI"
- "Video Encode with USB Camera"

5.4.4.4 Installing and Configuring GStreamer VAAPI

The GStreamer-VAAPI is a collection of GStreamer plugins and helper libraries that allow hardware-accelerated video decoding through VAAPI.

Note: Install libva and the Intel-VA driver before installing the GStreamer-VAAPI. Refer to “Installing VA Library” and “Installing and Configuring Video-based Preparation - Intel® Driver” on page 50.
For GStreamer-VAAPI support, follow the steps below:

1. Install GStreamer and plugins:
   
   ```
   # yum install gstreamer-devel gstreamer-plugins-base-devel
   gstreamer-plugins-base-tools gstreamer-plugins-plugins-base
   gstreamer-python gstreamer-rtsp gstreamer-plugins-bad-free-devel gstreamer-tools
   gstreamer-plugins-bad-free gstreamer-plugins-good gstreamer-plugins-ugly
   ```

2. Install GStreamer by checking out a copy of the source code of GStreamer-VAAPI. In the commands below, the 0.4.3 tag from the 0.4 branch of GStreamer VAAPI is used.
   
   ```
   # git clone git://gitorious.org/vaapi/gstreamer-vaapi.git
   # cd gstreamer-vaapi
   # git branch -a
   # git checkout remotes/origin/0.4-branch
   # git checkout 620cf3caa2169f8e09dcf7171c1dc098882b008c
   # git log
   ```

3. Move to the gstreamer-vaapi directory:
   
   ```
   # cd gstreamer-vaapi
   ```

4. Build the code and install it on your system:
   
   ```
   # ./autogen.sh
   # make
   # make install
   ```

**Note:** If `autogen.sh` fails to git clone gstreamer-codeparser, then you must git clone the codeparser and copy it. Codeparser can be obtained from the following link:

```
http://gitorious.org/vaapi/gstreamer-codecparsers/commits/gst-vaapi-branch
```

```
# cp -vrf gstreamer-codeparser/* vaapi-gstreamer-vaapi/ext/codeparser/
```

You may have to set the prefix to `/usr` if you want the library to be installed to `/usr`.

After the GStreamer-VAAPI is installed, different video formats can be played by launching `gst`.

### 5.4.4.4.1 VC-1 Interlace Video Decoding with Intel® EMGD

VC-1 Interlace video decoding is not currently supported by public OTC's GStreamer-VAAPI and the Intel driver solution. Therefore, VC-1 Interlace video decoding patches must be applied for the VC-1 frame interlace and field interlace function.

Perform the following steps to apply VC-1 video decoding patches for respective components.

**GStreamer-VAAPI**

1. Use git clone to check out gstreamer-codecparsers:
   
   ```
   # git clone git://gitorious.org/vaapi/gstreamer-codecparsers.git
   # git checkout 2e356b0f7efae33fb943ad11204020dcdbf1b04f
   # cd gstreamer-vaapi
   ```

2. Patch the GStreamer-VAAPI:
   
   ```
   # patch -p1 <
   Linux/patches/common/Gst_Plugins_VAAPI_pre_0_5_8_VC1_Interlace_Patch
   #./autogen.sh [options]
   ```
3. Enter the `ext/codecparsers` directory and apply the patch:
   
   ```
   # cd ext/codecparsers
   # patch -p1 <
   Linux/patches/common/Gst_Codecparsers_VAAPI_pre_0_5_8_VC1_Interlace_Patch
   ```

4. Build and install the code:

   ```
   # make
   # make install
   ```

### Intel® VA Driver

1. Use `git clone` to check out GStreamer-codecparsers:

   ```
   # git clone git://anongit.freedesktop.org/vaapi/intel-driver
   # git checkout 1.2.2
   ```

2. Patch the Intel®-VA driver:

   ```
   # patch -p1 <
   Linux/patches/common/Intel_VA_Driver_1_2_2_GEN7_VC1_Interlace_Patch
   ```

3. Build and install the code:

   ```
   # ./autogen.sh [options]
   # make
   # make install
   ```

#### Example 1. Video playback with audio: H.264 video format playback

   ```
   $gst-launch-1.0 filesrc location=<path_of_H264_video_sample>.mp4 ! qtdemux
   name=demux demux. ! queue2 ! vaapidecode ! vaapisink demux. ! aacparse ! queue2 !
   ffdec_aac ! alsasink
   ```

   ```
   $gst-launch-1.0 -v filesrc location= <path_of_H264_video_sample>.mpg !
   mpegtsdemux name=demuxer demuxer. ! queue vaapidecode ! vaapisink demuxer. !
   queue ! ffdec_mp3 ! audioconvert ! audioresample ! alsasink sync=true
   ```

#### Example 2. Video playback with audio: MPEG2 video format playback

   ```
   $gst-launch-1.0 -v filesrc location= <path_of_MPEG2_video_sample>.ts !
   mpegtsdemux name=demuxer demuxer. ! queue vaapidecode ! vaapisink demuxer. !
   queue ! ffdec_ac3 ! audioconvert ! audioresample ! alsasink sync=true
   ```

#### Example 3. Video playback with audio: VC1 video format playback

   ```
   $gst-launch-1.0 -v filesrc location= <path_of_VC-1_video_sample>.wmv ! asfdemux
   name=demuxer demuxer. ! queue vaapidecode ! vaapisink demuxer. ! queue !
   ffdec_wmapro ! audioconvert ! audioresample ! alsasink sync=true
   ```

#### Example 4. Video playback with audio: WMV video format playback

   ```
   $gst-launch-1.0 -v filesrc location= <path_of_WMV_video_sample>.wmv ! asfdemux
   name=demuxer demuxer. ! queue vaapidecode ! vaapisink demuxer. ! queue !
   ffdec_wmav2 ! audioconvert ! audioresample ! alsasink sync=true
   ```

#### Example 5. Video playback without audio: H.264 video format playback

   ```
   $gst-launch -v filesrc location=/<path_of_H264_video_sample>.mp4 ! qtdemux !
   vaapidecode ! vaapisink
   ```

   ```
   $gst-launch -v filesrc location=/<path_of_H264_video_sample>.mpg ! mpegtsdemux !
   vaapidecode ! vaapisink
   ```
Example 6. Video playback without audio: MPEG2 video format playback

$gst-launch -v filesrc location=/<path_of_MPEG2_video_sample>.ts ! mpegtsdemux ! vaapidecode ! vaapisink
$gst-launch-1.0 -v filesrc location=/<path_of_MPEG2_video_sample>.mpg ! mpegdemux ! vaapidecode ! vaapisink
$gst-launch-1.0 -v filesrc location=/<path_of_MPEG2_video_sample>.mpg ! mpegvideoparse ! vaapidecode ! vaapisink

Example 7. Video playback without audio: VC-1 video format playback

$gst-launch-1.0 -v filesrc location=/<path_of_VC-1_video_sample>.wmv ! asfdemux ! vaapidecode ! vaapisink

Example 8. Video playback without audio: WMV video format playback

$gst-launch-1.0 -v filesrc location=/<path_of_WMV_video_sample>.wmv ! asfdemux ! vaapidecode ! vaapisink

The GStreamer also has a media playing plugin that can handle playback with different formats.

$gst-launch-1.0 playbin uri=file:///home<File_name>

5.4.4.5 Video Encode with USB Camera

The VA-API (Video Acceleration API) is used to enable hardware accelerated video encode. Video encode in Linux* is supported up to standard definition at 30 fps.


2. The UVC (USB Video Class) must be enabled for the webcam to work. To check whether this feature is enabled, run the command:

   #vim .config

   Check whether the following configurations are included in your system:
   
   CONFIG_USB_VIDEO_CLASS
   CONFIG_MEDIA_USB_SUPPORT
   CONFIG_MEDIA_SUPPORT

   Note: If your kernel did not enable UVC, you must rebuild your kernel. Perform the following steps to configure the kernel.

   a. Go to the kernel source directory and configure it before the build:
      
      #make menuconfig

   b. Select the camera module you want to support:
      
      From the menu, select Device Drivers, Multimedia support (MEDIA_SUPPORT [=m]), Media USB Adapters (MEDIA_USB_SUPPORT [=y]), USB Video Class (UVC) (USB_VIDEO_CLASS [=m]), VIDEOBUF2_VMALLOC [=m]

   3. Rebuild the kernel and install it (refer to Appendix A).

4. Install V4L2 lib. This library is used to do the software color conversion. If you yum install libv4l later, you may need to recompile all GStreamer and plugins too.

   #yum install libv4l libv4l-devel

5. Install LibVA 1.2.1 (refer to Section 5.4.4.2 and Intel OTC’s VA driver 1.2.2 (refer to Section 5.4.4.3).
5.4.4.5.1 GStreamer 1.0 Base Preparation

The following steps describe the installation of required libraries to set up a working GStreamer-VAAPI plugins with GStreamer 1.2.x framework for the video encode. They are available to retrieve and upgrade by `yum` command.

1. Install GStreamer plugins:
   
   ```
   #yum install gstreamer-devel gstreamer-plugins-base-devel
   gstreamer-plugins-base-tools gstreamer-plugins-espeak gstreamer-plugins-base
   gstreamer-python gstreamer-rtsp gstreamer-plugins-bad-free-devel gstreamer-tools
   gstreamer-plugins-bad-free gstreamer-plugins-good gstreamer-plugins-ugly
   #yum install gstreamer1-devel gstreamer1-plugins-bad-free-devel
   gstreamer1-plugins-base-devel gettext-devel
   ```

   Currently the following gstreamer base and plugins version are used:
   - gstreamer1.i686 version 1.2.3 or 1.2.x
   - gstreamer1-plugins-base.i686 version 1.2.3 or 1.2.x
   - gstreamer1-plugins-bad-free.i686 version 1.2.3 or 1.2.x
   - gstreamer1-plugins-good.i686 version 1.2.3 or 1.2.x
   - gstreamer1-plugins-base-devel
   - gstreamer1-devel

2. Download `gst-plugins-ugly-1.2.3.tar.xz` or `1.2.x` (follow the based gstreamer1.i686 version) from
   [http://gstreamer.freedesktop.org/src/gst-plugins-ugly/gst-plugins-ugly-1.2.3.tar.xz](http://gstreamer.freedesktop.org/src/gst-plugins-ugly/gst-plugins-ugly-1.2.3.tar.xz)
   where x is the version that matches the other GStreamer libs version given by the distribution repository.

3. Untar the downloaded tar ball in the target system:
   ```
   #tar -xvJf <gst-plugin-ugly>.tar.xz
   ```

4. Build and install the code:
   ```
   #./autogen.sh
   #make
   #make install
   ```
5.4.4.5.2 GStreamer-VAAPI Video Encode

1. Check out a copy of the source code of Gstreamer-vaapi from:
   http://gitorious.org/vaapi/gstreamer-vaapi

   The master branch of Gstreamer VAAPI is used. Currently H264 and MPEG2 Video Encode works on Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processor N2807/N2930/J1900 platform.

   # git clone git://gitorious.org/vaapi/gstreamer-vaapi.git
   # cd gstreamer-vaapi
   # git checkout 2e356b0f7efae33fb943ad11204020dcdbf1b04f
   # git log (Shows commit logs)

2. Build and install the code:
   # cd gstreamer-vaapi
   # ./autogen.sh --enable-encoders
   # make
   # make install

Notes:
1) You may need to set the prefix to /usr if you want the library to be installed to /usr.

2) While running autogen.sh, if the system fails to git clone gstreamer-codeparser, then you must git clone codeparser and copy it. Here is the URL link:
   http://gitorious.org/vaapi/gstreamer-codecparsers/commits/gst-vaapi-branch

   # cp -vrf gstreamer-codeparser/* vaapi-gstreamer-vaapi/ext/codeparser/

To play video using the GStreamer-VAAPI plugins, use the following commands.
   #gst-launch-1.0 -v filesrc location=/path/to/video.mp4 ! qtdemux ! vaapidecode ! vaapisink
   OR
   #gst-launch-1.0 -v playbin uri=file:///root/tmp/output/test1.mpg

5.4.4.5.3 Video Encode Command Pipeline

1. USB Webcam command pipeline and display render output:
   — Command for MPEG2:
     $gst-launch-1.0 v4l2src device=/dev/video0 num-buffers=100 !
     video/x-raw,format=I420,width=640,height=480,framerate=30/1 ! tee name=enc
     ! queue ! vaapipostproc ! queue ! vaapiencode_mpeg2 ! mpegtsmux ! filesink
     location=2.mpg enc. ! queue ! vaapisink
   — Command for H264:
     $gst-launch-1.0 v4l2src device=/dev/video0 num-buffers=100 !
     video/x-raw,format=I420,width=640,height=480,framerate=30/1 ! tee name=enc
     ! queue ! vaapipostproc ! queue ! vaapiencode_h264 ! qtmux ! filesink
     location=2.mp4 enc. ! queue ! vaapisink

2. USB Webcam command pipeline:
   — Command for MPEG2:
     $gst-launch-1.0 v4l2src device=/dev/video0 num-buffers=200 !
     video/x-raw,format=I420,width=640,height=1080♭ vaapipostproc ! queue !
     vaapiencode_mpeg2 ! mpegtsmux ! filesink location=test2.mp4
   — Command for H264:
     $gst-launch-1.0 v4l2src device=/dev/video0 num-buffers=200 !
     video/x-raw,format=YUY2,width=1920,height=1080♭ vaapipostproc ! queue !
     vaapiencode_h264 ! qtmux ! filesink location=test2.mp4
3. Encode through raw video file:
   — Command for MPEG2:
     ```
     $gst-launch-1.0 filesrc location=/root/tmp/videoRawFile/H264_NV12.yuv !
     videoparse format=nv12 width=1280 height=720 framerate=30/1 ! vaapipostproc
     ! queue ! vaapiencode_mpeg2 ! mpegtsmux ! filesink location=test2.mpg
     ```
   — Command for H264:
     ```
     $gst-launch-1.0 filesrc location=/root/tmp/videoRawFile/H264_NV12.yuv !
     videoparse format=nv12 width=1280 height=720 framerate=30/1 ! vaapipostproc
     ! queue ! vaapiencode_h264 ! qtmux ! filesink location=test2.mp4
     ```

4. Command pipeline for using Raw file to encode and render the encode:
   ```
   $gst-launch-1.0 filesrc
   location=/root/tmp/videoRawFile/YUY2_Prog_Packed_720x480.yuv !
   videoparse,format=yuy2,width=720,height=480,framerate=30/1 ! tee name=enc
   queue ! vaapipostproc ! queue ! vaapiencode_mpeg2 ! mpegtsmux ! filesink
   location=2.mpg enc. ! queue ! vaapisink
   ```

5. Command pipeline for using USB camera to encode and render the encode:
   ```
   $gst-launch-1.0 v4l2src device=/dev/video0 num-buffers=100 !
   video/x-raw,format=I420,width=640,height=480,framerate=30/1 ! tee name=enc
   queue ! vaapipostproc ! queue ! vaapiencode_mpeg2 ! mpegtsmux ! filesink
   location=2.mpg enc. ! queue ! vaapisink
   ```

See also:
- “Hardware Video Acceleration (VA) Overview”
- “Installing VA Library”
- “Installing and Configuring Video-based Preparation - Intel® Driver”

5.5 Configuring the Intel® EMGD on Linux* OS Using CED Lite and Generating Binary Blob

Intel® EMGD auto-detects all device information necessary to initialize the integrated graphics device in most configurations. This section only covers configuration details specific to Intel® EMGD.

To configure Intel® EMGD for Linux*, you can use CED Lite.

5.5.1 Linux* OS Configuration Using CED Lite

Intel® EMGD for Linux* comes with a tool that allows users to configure the drive to suit their needs through the "binary blob," without having to rebuild the entire Intel® EMGD with CED Lite. This tool is called CED Lite and is an OS-independent browser-based application that can be found in your EMGD/CED-Lite-Lite folder.
If you experience any problems, ensure you are using the latest updated version of Firefox®. If you use Firefox®, ensure that you are using Firefox® 15 or later for the proper functioning of CED Lite.

Note: Refer to Section 3.0, “Configuration and CED Lite” on page 15 for more information about using the tool.

The following steps and screen shots outline a typical configuration file generation procedure using CED Lite. Note that these steps are for reference only; your configuration may differ.

See also:
- “Prerequisites for CED Lite”
- “Creating the Configuration”
- “Generating and Installing the Binary Blob”
5.5.2 Prerequisites for CED Lite

- Python 2.7 on your host (to launch CED Lite)
- Firefox* version 15 or higher
- Set Firefox* version 15 as the default web browser to ensure full compatibility

*Note:* To set Firefox* as your default browser, open Firefox*, from the menu bar click Tools, Options, select the Advanced tab, and then select Make Firefox the default browser.

5.5.3 Creating the Configuration

*Note:* This example illustrates the configuration on Dual-HDMI.

1. To launch CED Lite from the command line, change to the CED Lite home directory.
2. Execute the command python launch.py. CED Lite opens in the system's default browser.
3. Select Clone display mode.
4. Under Primary Display, choose HDMI and Connector B.
5. Under Secondary Display, choose HDMI and Connector C.
6. Display (Advanced Settings) allows users to set the EDID option, specify the port name (2 = HDMI-B, 3 = HDMI-C, 5 = VGA, 6 = DP-B and 7 = DP-C / edp), display timing resolution, etc. Example: Setup on Dual-HDMI is with EDID 111, Port Name set to 2.

7. Under the EDID options, select **Advanced** and check the **Use driver built-in standard** check box, leaving the rest unchecked.
8. Apply similar settings for the secondary display (HDMI-C).

![HDMI Display (Advanced Settings)](image)

**Note:** The Display (Advanced setting) option changes based on the display mode selection and the primary/secondary display selection of the user.

9. After the desired setting changes are completed, click **Generate Intel EMGD Configuration** to generate three files: `image_data.txt`, `xorg.txt`, and `user_config.txt` located at CED-Lite\Installation\Linux folder. These files can be used for generating the “binary blob.”

You will then receive three prompts:
- File `image_data.txt` saved successfully - click **OK**.
- File `xorg.txt` saved successfully - click **OK**.
- File `user_config.txt` saved successfully - click **OK**.

### 5.5.4 Generating and Installing the Binary Blob

1. Go to the EMGD driver package, Linux* directory and untar the subsequent files using the following commands:
   ```bash
   # cd Linux
   # tar-zxvf IEMGD_HEAD_Linux.tgz
   ```

2. After `IEMGD_HEAD_Linux.tgz` is extracted, change to the sub directory `common/drm` and untar `emgd_drm.tgz` package and enter the `tools/fw_tool` subdirectory.
   ```bash
   # cd common/drm/
   # tar -zxvf emgd_drm.tgz
   # cd tools/fw_tool
   ```
3. Copy the `user_config.txt` file into the `fw_tool/` directory and rename it as `user_config.c`
   
   ```bash
   #cp <Path to EMGD package>/CED-Lite/installation/Linux/user_config.txt ./
   #mv user_config.txt user_config.c
   ```

4. Copy the `image_date.txt` file into the `fw_tool/` directory and rename it as `image_data.h`
   
   ```bash
   #cp <Path to EMGD package>/CED-Lite/installation/Linux/image_data.txt ./
   #mv image_data.txt image_data.h
   ```

5. Execute the following command lines to build the `emgd.bin`:
   
   ```bash
   #make clean
   #make fw_tool
   #./fw_tool
   ```

   **Note:** Make sure that no error occurs and that you see the message below:
   
   `Opened output file emgd_32.bin.`

6. Copy the generated `emgd.bin` file to `/lib/firmware` directory:
   
   ```bash
   #cp emgd_32.bin /lib/firmware
   ```

7. Execute the following command to load:
   
   ```bash
   #modprobe emgd firmware="emgd_32.bin"
   ```

8. On linux load line in file `/etc/default/grub`, add to GRUB_CMDLINE_LINUX at the end:
   
   ```bash
   emgd.firmware=emgd_32.bin
   ```

9. Remake grub by first logging in as root. Then type the following:
   
   ```bash
   #grub2-mkconfig -o /boot/efi/EFI/fedora/grub.cfg
   ```

   **Note:** To load with the default configuration, use `modprobe emgd`.

### 5.5.5 Generating the Configuration Files with the Firmware Binary Blob

The data contained in the `emgd.bin` can be used to create configuration files using the reverse firmware tool.

1. Go to the EMGD driver package, `Linux` directory and untar the subsequent files using the below commands:
   
   ```bash
   #cd Linux
   #tar -zxvf IEMGD_HEAD_Linux.tgz
   ```

2. After `IEMGD_HEAD_Linux.tgz` is extracted, change to the sub directory `common/drm` and untar `emgd_drm.tgz` package and enter the `tools/fw_tool` subdirectory.
   
   ```bash
   #cd common/drm/
   #tar -zxvf emgd_drm.tgz
   #cd tools/fw_tool
   ```

3. Copy the input file `emgd_64.bin` (for 64 bit) or `emgd_32.bin` (for 32bit) into the `fw_tool` directory
   
   ```bash
   #cp /lib/firmware/emgd_64.bin ./
   ```

4. Execute the following command lines to create `new_user_config.c` and `new_image_data.h`, representing the data contained in the `emgd.bin` file.
   
   ```bash
   #make clean
   #make rev-fw_tool
   #./rev-fw_tool
   ```
5.6 The Linux* OS Configuration File

To configure Intel® EMGD for use with Linux*, there are several sections within the Linux* configuration file (xorg.conf) that must be edited or created, including:

- "Device Section"
- "Screen Section"
- "ServerLayout Section"

The above Sections are described in following the example files. See the xorg.conf man pages for complete details.

Example 9. xorg.conf File for Fedora* 18

```
##
## X Config options generated from CED
## x11 conf skeleton
## DriverVer=
##
## Section "Screen"
Identifier    "Screen0"
Device        "IntelEMGD-0"
Monitor       "Monitor0"
SubSection    "Display"
EndSubSection
EndSection

# Primary (First/only) display
Section "Device"
  Identifier "IntelEMGD-0"
  Driver     "emgd"
  VendorName "Intel(R) DEG"
  BoardName  "Embedded Graphics"
  BusID      "0:2:0"
  Screen     0
  Option     "PcfVersion"            "1792"
  Option     "ConfigId"              "1"
  Option     "ALL/1/Port/5/General/name"           "vga1"
  Option     "ALL/1/Port/5/General/Rotate"         "normal"
  Option     "ALL/1/Port/2/General/name"           "hdmi2"
  Option     "ALL/1/Port/2/General/Rotate"         "normal"
EndSection

Section "ServerLayout"
  Identifier     "Default Layout"
  Screen 0       "Screen0" 0 0
  # InputDevice    "Mouse0" "CorePointer"
  # InputDevice    "Keyboard0" "CoreKeyboard"
  # InputDevice    "DevInputMice" "SendCoreEvents"
EndSection
```

5.6.1 Device Section

The Device section provides a description of a graphics device. The Linux* configuration file (xorg.conf) must have at least one Device section for the graphics driver. If your chipset supports multiple graphics pipelines, you may have multiple Device sections, but in most situations, only one is required. Device sections in xorg.conf have the following format:
Section “Device”
  Identifier “devname”
  Driver “emgd”
...
EndSection

The Identifier field defines the device. This name associates the device with a screen in the Screen sections.

The Driver field defines the driver to use and is a required field in the Device section. The Intel driver, emgd_drv.o, must be installed in the /usr/lib/xorg/modules/drivers (or the correct path for your system).

5.6.2 Screen Section

The Screen section binds a Screen with a Driver and a Monitor. It defines resolution modes, color depths, and various other screen characteristics. See the xorg man page for detailed information.

The Screen section has the following format:

Section "Screen"
  Identifier “screenname”
  Device “devname”
  Monitor “Monitor0”
  Subsection “Display”
EndSubSection

EndSection

5.6.3 ServerLayout Section

The ServerLayout section defines the overall layout of the system configuration. Input devices are specified in the InputDevice fields and output devices usually consist of multiple components, such as a graphics board and a monitor, which are bound together in a Screen section.

Section "ServerLayout"
  Identifier “Default Layout”
  Screen 0 “Screen0” 0 0
  # InputDevice “Mouse0” “CorePointer”
  # InputDevice “Keyboard0” “CoreKeyboard”
  # InputDevice “DevInputMice” “SendCoreEvents”

EndSection

Note: Refer to Appendix C for details on the user_config.c file.

5.7 Runtime Configuration GUI

You can change the configuration and runtime attributes of the driver using the emgdgui runtime configuration tool. The Intel® EMGD GUI (emgdgui) is an application that is used to view and control Intel® EMGD. It retrieves the status of the display and driver and is also used to configure the supported display attributes. You can change the configuration and runtime attributes of the driver using the emgdgui runtime configuration tool, which resides in the <EMGD driver package>/Linux/Utilities directory. The emgdgui also demonstrates multi-monitor support. Note that changes made with emgdgui are not permanent and are lost upon rebooting.
See also:
- "Emgdgui Setup"
- "Using the emgdgui Runtime Configuration Utility"

5.7.1 Emgdgui Setup

To run `emgdgui`, you must ensure that the X server has been configured to use Intel® EMGD. See “Configuring the Intel® EMGD on Linux* OS Using CED Lite and Generating Binary Blob” on page 56 for details on configuring and installing the Intel® EMGD.

1. The `emgdgui` is available in the `<EMGD driver package>/Linux/Utilities` directory.
2. Untar the `emgdgui.tgz` file.
   `$tar -xvzf emgdgui.tgz`
3. Change the directory to `emgdgui`.
   `$cd emgdgui`
4. Execute the "emgdguid" binary.
   `$./emgduid`
5. Open the browser (Firefox* or Chrome*) and set the URL to:
   `http://localhost:8888`
   The `emgdgui` will open in the web browser.

5.7.2 Using the emgdgui Runtime Configuration Utility

The `emgdgui` application provides five tabs: Driver Info, CRTCs/Pipes, Planes, Modeset, and Color Correction.

- **Driver Info**: Contains the driver information.
- **CRTCs/Pipes**: Contains the current display mode such as the CRT timings and information for framebuffer. Using this tab, you can also set the Z-order of the sprite planes.
- **Planes**: Contains color-correction settings for the sprite plane that are different from the main framebuffer color-correction settings. This feature lets you color-correct for red, green, and blue, plus it enables you to adjust brightness, contrast, and saturation.
- **Modeset**: Contains the desired display modes and the resolutions for the respective displays connected to the system.
- **Color Correction**: Contains current color-correction information for the respective display selected.
Figure 3 below shows the Driver Info tab.

**Figure 3. Example Runtime Configuration GUI - Driver Info Tab**

![Driver Info Tab](image)

The CRTC/Pipes tab enables you to view the current display mode information, framebuffer information, or change the overlay plane Z-order.

*Note:* If you make any changes to the configuration, click **Apply** for the changes to take effect.
The following steps present an example planes configuration procedure:

1. Select Plane Config for Sprite-A.
2. Select the required color to be corrected in the Gamma section.
3. Select the required color attribute to be corrected in the Other correction section.
4. Similarly make the desired configurations for Sprite-B and Sprite-C.
5. Click Apply plane configurations for changes to take effect or click Defaults to restore the default configuration.
Figure 5. Example Runtime Configuration GUI - Planes Tab
The Modeset dialog box allows the user to choose the primary and the secondary displays. You can change display resolutions using the Resolution drop-down list.

Figure 6. Example Runtime Configuration GUI - Modeset Tab
The following steps present an example color-correction procedure:

1. Select the required color attribute to be corrected in the Gamma Correction section.
2. Select the required color attribute to be corrected in the Brightness Correction section.
3. Select the required color attribute to be corrected in the Contrast Correction section.
4. Click **Apply** to save your changes or **Defaults** to restore the default values.
5.8 Hue Control Support for Sprite

You can pre-configure the hue through `user_config.c`.

To configure the hue value for Sprite:
- Set the hue value directly through `sysfs` (see Section 5.10).
- Set the hue in `user_config.c`:

```c
emgd_ovl_config_t initial_ovl_config = {
    {
        /* igd_ovl_gamma_info */
        /***************************************************************************/
        0x00000100,/* unsigned int red; */
        0x00000100,/* unsigned int green; */
        0x00000100,/* unsigned int blue; */
        IGD_OVL_GAMMA_DISABLE /* unsigned int flags; */
    },
    {
        /* igd_ovl_color_correct_info_t */
        /***************************************************************************/
        -5,/* short brightness; */
        67,/* unsigned short contrast; */
        145,/* unsigned short saturation; */
        0/* unsigned short hue; */
    }
},
```

The hue must be set for all four sprites.

*Note:* The above example is generated with the default configuration.

5.9 Xrandr

`xrandr` is used to set the size, orientation and/or reflection of the outputs for a screen. It can also set the screen size. `xrandr` is a standard Linux* capability. Intel® EMGD supports `xrandr`. There are a few global options; the rest modify a particular output and follow the specification of that output on the command line.
The following are examples for switching the display mode and screen rotation using `xrandr`:

**Switching display mode:**

**Single (to turn off another display):**
```
xrandr --output <primary port name> --auto --mode 1024x768 -r 60 --output <secondary port name> --off
```

**Extended:**
```
xrandr --output <primary port-name> --auto -mode 1024x768 -r 60 --output <secondary port-name> --mode 1024x768 -r 60 --right-of <primary port-name>
```

**Vertical Extended (below):**
```
xrandr --output <primary port-name> --auto --mode 1024x768 -r 60 --output <secondary port-name> --auto --mode 1024x768 -r 60 --below <primary port-name>
```

**Vertical Extended (above):**
```
xrandr --output <primary port-name> --auto --mode 1024x768 -r 60 --output <secondary port-name> --auto --mode 1024x768 -r 60 --above <primary port-name>
```

**Clone:**
```
xrandr --output <primary port-name> --auto --mode 1024x768 -r 60 --output <secondary port-name> --mode 1024x768 -r 60 --same-as <primary port-name>
```

**Secondary clone:**
```
xrandr --output <secondary port-name> --auto --mode 1024x768 -r 60 --output <primary port-name> --mode 1024x768 -r 60 --same-as <secondary port-name>
```

**Rotation:**

**90 degree:**
```
xrandr --output <port-name> --auto --mode 1024x768 -r 60 --rotate left
```

**180 degree**
```
xrandr --output <port-name> --auto --mode 1024x768 -r 60 --rotate inverted
```

**270 degree:**
```
xrandr --output <port-name> --auto --mode 1024x768 -r 60 --rotate right
```

### 5.10 Intel® EMGD Custom Control Using the sysfs Interface

The Intel® EMGD kernel driver initializes nodes in sysfs to control a number of custom driver functions. These nodes are located under:
```
/sys/module/emgd
```

The sysfs nodes may be accessed from the command line using the `ls`, `cat`, and `echo` commands. Use `cat` to query and `echo` to edit a value.
Examples

To lists all the nodes for a specific crtc:

```
ls /sys/module/emgd/control/crtc0/
```

To see the status of a crtc plane:

```
cat /sys/modules/emgd/control/crtc0/lock_plane
```

To set framebuffer brightness to full (example here is HDMIA port):

```
echo "0xffffffff" > /sys/module/emgd/control/HDMI-A-1/brightness
```

Sprite attributes:

- Sprite A: /sys/module/emgd/control/PlaneId1/
- Sprite B: /sys/module/emgd/control/PlaneId2/
- Sprite C: /sys/module/emgd/control/PlaneId3/
- Sprite D: /sys/module/emgd/control/PlaneId4/

Display Plane attributes:

- Display Plane 1: /sys/module/emgd/control/crtc0/
- Display Plane 2: /sys/module/emgd/control/crtc1/

Display Port attributes:

- VGA port: /sys/module/emgd/control/VGA-1/
- HDMI A port: /sys/module/emgd/control/HDMI-A-1/
- DP A port: /sys/module/emgd/control/DP-1/
- DP B port: /sys/module/emgd/control/DP-2/

EMGD info:

```
/sys/module/emgd/info/version
/sys/module/emgd/info/description
/sys/module/emgd/info/chipset
```

Examples:

To freeze display plane:

```
echo "1" > /sys/module/emgd/control/crtc0/lock_plane
```

To enable DPMS:

```
echo "1" /sys/module/emgd/control/crtc0/DPMS
```

To enable fb_BlendOvl:

```
echo "1" /sys/module/emgd/control/crtc0/fb_blend_ovl
```

To check current mode:

```
cat /sys/module/emgd/control/crtc0/current_mode
```

To write or read hue via sysfs:

```
echo [value_between_0x0000_and_0xFFFF] > /sys/modules/emgd/control/PlaneId[1~4]/hue
```

This will set a value between 0x0000 and 0xFFFF of hue to Sprite A~D (1~4).

To read the current set value:

```
cat /sys/modules/emgd/control/PlaneId[1~4]/hue
```
5.11 Sprite Performance through VAAPi Stack with VAXV Support

Intel® Atom™ Processor E3800 Product Family/Intel® Celeron® Processor N2807/N2930/J1900 have sprite with YUV format support. Through VAXV, a user media player can extend the functionality and make full use of the sprite capability. With YUV and VAXV support, the video image presentation can greatly improve the GPU performance through VAAPi stack.

5.11.1 Enable Sprite in VAAPi Stack with VAXV Support

This approach is for working under the X-Window system. To enable this feature, the user must patch the video driver with VAXV support. The patches can be found in this package. Intel® EMGD’s DDX driver is an open source XOrg server part of this package and VAXV support is already included. Intel® EMGD reuses OTC’s video driver. To add VAXV support, you must patch the video driver with extra patches that are included in this package. (Refer to Section 5.4.4.3 for basic video driver build and install instruction). Currently, Intel® EMGD provides patches for video driver version 1.2.0 and 1.2.1 (the same patch works on both of these versions).

After patching with VAXV support, you can start building the video driver. The VAXV components are depended on the libXv development package (libXv-devel). After that development package is installed, the “xv” is enabled as one of the windowing systems at the end of the autogen.sh script, similar to the following:

```
autoreconf: Entering directory `.
autoreconf: configure.ac: not using Gettext
...
... Windowing systems ................ : drm x11 xv wayland
```

5.11.2 Tools to Test the Feature

The GStreamer-VAAPi or MPlayer-VAAPi can be used to test the feature. The patches for both of these tools are included in the packages. The patches are for GStreamer-VAAPi version 0.5.8 and MPlayer-VAAPi version.

To test on GStreamer-VAAPi, add an additional option after vaapisink,

```
render-mode= overlay
```

Example:
```
#gst-launch-1.0 filesrc location=<video clip> ! <demuxer> ! vaapidecode ! vaapisink
render-mode=overlay
```

See also:
- “Entering Pipe Settings”
- “Sprite Plane Settings”
- “Flipping an OGL Surface onto Sprite Using Xprop”
5.12 Flipping an OGL Surface onto Sprite Using Xprop

The xprop utility is used for displaying window and font properties in an X server. One window or font is selected using the command line arguments or possibly in the case of a window, by clicking on the desired window.

```
xprop –id window_id –format EMGD_USE_SPRITE 8s –set EMGD_USE_SPRITE foo;
```

(where `foo` can be any non-zero value)

See also:
- “Entering Pipe Settings”
- “Sprite Plane Settings”
- "Sprite Performance through VAAPI Stack with VAXV Support”

5.13 Enabling HDMI Audio

The Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processor N2807/N2930/J1900 platform supports HDMI audio. Use the following steps to enable HDMI audio.

1. Enter system BIOS and change the configuration as follows.
   Go to South Cluster Configuration- Audio Configuration - Audio Controller (Enable)- Make sure HDMI Codec is enabled.
2. Save the settings and boot into the operating system.
3. To check whether the ALSA audio driver is recognized, list all the devices that the ALSA audio driver recognized by typing the following command in the terminal:
   ```sh
   $aplay –l
   ```
   The results should contain the devices shown in the following example.

   **Example 10. Recognized Audio Devices**
   ```
   — card 0: PCH [HDA Intel PCH], device 0: ALC889A Analog [ALC889A Analog] Subdevices: 1/1
   — Subdevice #0: subdevice #0
   — card 0: PCH [HDA Intel PCH], device 1: ALC889A Digital [ALC889A Digital]
   — Subdevices: 1/1
   — Subdevice #0: subdevice #0
   — card 0: PCH [HDA Intel PCH], device 3: HDMI 0 [HDMI 0]
   — Subdevices: 1/1
   — Subdevice #0: subdevice #0
   — card 0: PCH [HDA Intel PCH], device 7: HDMI 1 [HDMI 1]
   — Subdevices: 1/1
   — Subdevice #0: subdevice #0
   — card 0: PCH [HDA Intel PCH], device 8: HDMI 2 [HDMI 2]
   — Subdevices: 1/1
   — Subdevice #0: subdevice #0
   ```
4. To get the sound output from the HDMI monitor through the HDMI cable, the correct audio output device must be specified. For example:
   ```sh
   $./mplayer –va vaapi –vo vaapi –ao alsa:device=hw=X.Y video_file_name
   ```
   where X=card, Y=device
5.14 Enabling DisplayPort Audio

The Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processor N2807/ N2930/J1900 platform supports DP audio.

Use the following steps to enable DP audio.

1. Enter the system BIOS and change the configuration as follows.
   - Go to South Cluster Configuration - Audio Configuration - Audio Controller (Enable) - Make sure HDMI Codec is enabled.
2. Save the settings and boot into the operating system.
3. To check whether the ALSA audio driver is recognized, list all the devices that the ALSA audio driver recognized by typing the following command at the terminal:
   ```bash
   $aplay –l
   ```
4. To get the sound output from the DP monitor through the DP cable, the correct audio output device must be specified.
   - For example:
     ```bash
     ./mplayer –va vaapi –vo vaapi –ao alsa:device=hw=X.Y video_file_name
     ```
     where X=card, Y=device

5.15 RC6 Feature

The Intel® RC6 feature allows the Graphics Processing Unit (GPU) to enter a lower power state during GPU idle.

1. There are two types of RC6 counters, Render RC6 counter (0x138108) and Media RC6 counter (0x13810).
   a. Render RC6 counter only triggers when running a render application.
   b. The Render/Media Rc6 starts to trigger when Media application is played.

Note: The EMGD-DRM RC6 render counter and RC6 media counter starts to trigger even if you are using only 3D rendering.

Verification for RC6 for Intel® EMGD:

   a. Get regmon (contact your Intel representative) or Intel-gpu-tools, but you must add 0x180000 offset
   b. Execute ./regmon -device:i915 -read -offset:0x138108 (render RC6 (standby counter))
   c. Observation:
      - Notice that the Render/Media Rc6 counter starts to count if there is no rendering/video application launch
      - When the application is launched, the Rc6 counters stop counting.
5.16  EMGD S3D Modes

The Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processor N2807/ N2930/J1900 platform supports Stereoscopic 3D. Stereoscopic (3-D imaging) is a technique capable of recording three-dimensional visual information or creating the illusion of depth in an image. Stereoscopic is the enhancement of the illusion of depth in photograph, movie, or other two-dimensional image by presenting a slightly different image to each eye. Many 3D displays use this method to convey images.

The methods describes S3D modes setting with a single pre-combined S3D buffer. Single pre-combined S3D buffer refers to a buffer created by application with the appropriate Left and Right picture arrangement in the buffer as required by a specific 3D format.

Mode Setting Method 1:
1. Find the connector with "expose 3D modes" and "S3D Set" properties.
2. Set "expose 3D modes" property to 0 to disable expose 3D modes. Expose 3D modes is used for setting 3D modes in Method 2. Method 2 always supersedes Method 1 if enable, so it is always safer to first disable expose 3D modes.
3. Prepare a buffer with the appropriate 3D content for the expected 3D format listed in Table 20.
4. Call drmModeConnectorSetProperty to set "S3D set" connector property to the expected 3D format. This must be done before calling drmModeSetCrtc.
5. Call drmModeSetCrtc to set the expected 3D mode.

Notes:
- The only reason "expose 3D modes" property is used here is to ensure it is disabled.
- Any changes to "S3D set" connector property only take effect in the next mode set.
- This method can be used to set any valid modes in the connector mode list to a 3D mode as listed in the table below. However, the output on the monitor may not exhibit 3D behavior if the monitor does not support stereo 3D or does not support certain 3D formats/modes.

<table>
<thead>
<tr>
<th>Value Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PD_S3D_MODE_OFF</td>
<td>0x00</td>
<td>(default) Disable Stereo 3D</td>
</tr>
<tr>
<td>PD_S3D_MODE_FRAME_PACKING</td>
<td>0x10</td>
<td>Frame-packing format</td>
</tr>
</tbody>
</table>
| PD_S3D_MODE_SBS_HALF    | 0x20   | Side-by-side Half format with any one of the following sub-sampling. Example: 0x22 represents Side-by-side Half format with horizontal sub-sampling Even/Left picture and Odd/Right picture \[nn\] Sub-sampling:
  - 0x00  Horizontal Sub-sampling, Odd/Left, Odd/Right
  - 0x01  Horizontal Sub-sampling, Odd/Left, Even/Right
  - 0x02  Horizontal Sub-sampling, Even/Left, Odd/Right
  - 0x03  Horizontal Sub-sampling, Even/Left, Even/Right
  - 0x04  Quincunx matrix, Odd/Left, Odd/Right
  - 0x05  Quincunx matrix, Odd/Left, Even/Right
  - 0x06  Quincunx matrix, Even/Left, Odd/Right
  - 0x07  Quincunx matrix, Even/Left, Even/Right
| PD_S3D_MODE_TOP_BOTTOM  | 0x30   | Top-and-Bottom format                                     |
| PD_S3D_MODE_SBS_FULL    | 0x40   | Side-by-side Full format                                  |
**Mode Setting Method 2:**

1. Find the connector with the "expose 3D modes" property.
2. Set the "expose 3D modes" connector property to 1.
3. Call drmModeGetConnector to refresh the mode list in the connector. If 3D monitor attached, modes that match the mandatory 3D modes will be marked with any combination of DRM_MODE_FLAG shown below, if applicable:
   - DRM_MODE_FLAG_3D_TOP_BOTTOM
   - DRM_MODE_FLAG_3D_SIDE_BY_SIDE_HALF
   - DRM_MODE_FLAG_3D_FRAME_PACKING
4. Prepare a buffer with the appropriate 3D content for set mode.
5. Call drmModeSetCrtc with mode->flags set to an expected 3D format corresponding to one of the DRM_MODE_FLAG to set the 3D mode.

**Notes:**
- This method requires a 3D monitor with EDID content that indicates 3D support to work correctly.
- In this method, drmModeSetCrtc can be called to set any valid modes in the connector mode list to a 3D mode listed in Table 21, even though the mode->flags do not indicate that it supports certain 3D format. However, the output on the monitor may not exhibit correct 3D behavior if the 3D monitor does not support the requested 3D formats in that mode.

**Table 21. "expose 3D modes" Connector Property**

<table>
<thead>
<tr>
<th>Value Name</th>
<th>Value</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>PD_S3D_MODE_EXPOSE_OFF</td>
<td>0x00</td>
<td>(default) Disable &quot;expose 3D modes&quot;.</td>
</tr>
<tr>
<td>PD_S3D_MODE_EXPOSE_ON</td>
<td>0x01</td>
<td>Enable &quot;expose 3D modes&quot;. The follow mode flags will be exposed if display support S3D: DRM_MODE_FLAG_3D_TOP_BOTTOM DRM_MODE_FLAG_3D_SIDE_BY_SIDE_HALF DRM_MODE_FLAG_3D_FRAME_PACKING</td>
</tr>
</tbody>
</table>

**5.17 Enabling Hot-plug**

The driver by default has no hotplug enabled. Hotplug feature can be enabled by manually adding the hotplug flag in the default configuration file or using CED lite.

**Default configuration file**

- IGD_DISPLAY_MULTI_DVO | IGD_DISPLAY_DETECT,
- IGD_DISPLAY_HOTPLUG | IGD_DISPLAY_MULTI_DVO | IGD_DISPLAY_DETECT,

**CED Lite**

- Enable display hotplug flag.
- No bios setting for hotplug is needed.
Appendix A Fedora* 18 Bring-up with Kernel 3.8 on Intel® E3800/N2807/N2930/J1900

Note: The following set of steps for creating Fedora* Boot images is for reference only. Intel does not guarantee it will work on your specific environment. You are encouraged to use the Timesys* distributed Fedora* Linux* BSP recommended by Intel.

A.1 Prerequisites
The following lists the prerequisites for installing and configuring Intel® EMGD for Linux*:

• Development Platform with supported Intel chipset
• Hard disk drive to be used on Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processor N2807/N2930/J1900 platform
• Linux* kernel header package for active running kernel
  — Direct Rendering support enabled
• USB drive
• Intel® Atom™ Processor E3800 Product Family/ Intel® Celeron® Processor N2807/N2930/J1900 platform
• System administration privileges
• Internet connection

A.2 Steps
2. Create a LIVE bootable USB device using software, named Live USB Creator. Install Fedora* 18 into the HARD DISK DRIVE.

See also:
• “Installing Live Bootable Image into the Hard Disk Drive”
• “Update Fedora* 18 to Latest Kernel Version”

A.3 Installing Live Bootable Image into the Hard Disk Drive
1. Plug in the USB and hard disk drive to a host machine and boot to USB. You should get a live version of Fedora* 18.
2. Choose the Troubleshooting, **Start Fedora with basic graphics** option in case your machine is unable to boot GNOME graphics.

3. Click **Live System User**. No password is needed.
4. Choose **Install to Hard Drive** to install Fedora 18 onto your hard disk drive.
5. Select your language and then click **Continue**.
6. Set the date and time and then click **Installation Destination**.

7. Choose the hard disk drive ion which you want to install Fedora* 18 and then click **Continue**.

8. In the partition scheme configuration, choose **Standard Partition Type** and then **Continue**.
9. If the system displays a prompt indicating that your hard disk drive does not have enough space to install Fedora®, click **Reclaim space**.

10. Select each of the partition and then click **Delete** to wipe out the partition. It is better to delete everything and then click **Reclaim Disk Space**. Then, click **DONE** to return to the Installation Summary Menu.
11. After the previous steps are completed, you are ready to install. Click **Begin Installations**. The installations takes 15 - 20 minutes depending on the frequency of your machine and speed of your hard disk drive. Meanwhile, you can set your root (super user) password during the installation process.
A.4 Update Fedora* 18 to Latest Kernel Version

1. Boot Fedora* 18 using your hard disk drive. You will be prompted to set up a User Account during start-up.

2. Click Forward until you reach the Create User page.

4. Set your username and user password.

Note: This password is the password you will use to log on your system; it is not the same as the root password you set previously.

5. After you log on into your system, in System Settings, click Details and then Graphics. Turn on the Forced Fallback Mode. Reboot your machine and you will get the Fallback Mode GNOME desktop.
6. In System Settings, click **Network** to set the network proxy.

![Network Proxy Interface](image)

7. Update the kernel and development package, running the following commands as root user to update your system:
   ```shell
   # yum update -y
   # yum install kernel-headers kernel-devel ncurses-devel make gcc patch -y
   ```

8. After the update is complete, reboot your system to enable the update.

9. Untar the Linux-3.8 vanilla kernel source:
   ```shell
   # tar -xvzf linux-3.8.tar.gz
   ```

10. Untar the Linux 3.8 kernel patches:
    ```shell
    # tar -xvjf <patch version>.tar.bz2
    ```

    **Note:** The following steps require Intel® Atom™ Processor E3800 Product Family/Intel® Celeron® Processor N2807/N2930/J1900 driver patches from Intel. However, these steps are not required with the Timesys® BSP because it is already patched.

11. Enter the vanilla kernel source folder and apply all patches.
    ```shell
    # cd linux-3.8/
    # patch -p1 < /<Path-to-patch>/LSP_Patches/AV_LSP_3.8_<Device>_<date>.patch
    ```
    **Example:**
    ```shell
    # patch -p1 < ../LSP_Patches/AV_LSP_3.8_ACPI_2013-04-26.patch
    # patch -p1 < ../LSP_Patches/AV_LSP_3.8_CLK_2013-04-26.patch
    ```
    Ensure that all patches are applied properly.

12. Enter the kernel directory and copy the existing configuration file to the kernel top directory:
    ```shell
    # cd linux-3.8/
    # cp -vrf arch/x86/configs/<name of the config file> ./.config
    ```
13. Build the kernel:
   #make ARCH=i386
14. After the compilation is done, install the kernel into the system:
   #make modules_install
   #make install
   The kernel will update grub.cfg automatically.
15. Check that the default menu entry is pointed to the menu entry of your custom build kernel.
   In /boot/grub2/grub.cfg, make sure you set default to 0.
16. Reboot the platform and it will boot to your custom built kernel.
Appendix B 2D/3D API Support

The OpenGL implementation for Intel® EMGD consists of three components.

- **libGL**: This is the shared library that implements the OpenGL and GLX APIs. It is linked by all OpenGL applications.
- **emgd.ko**: This is the Direct Rendering Manager (DRM). It is a kernel module that provides the OpenGL application with the permissions necessary to directly access the DMA buffers used by libGL.
- **X server**: The existing Intel® EMGD X server driver has been enhanced to communicate with libGL.

Installing the Intel® EMGD OpenGL driver provides a fully hardware accelerated implementation of the OpenGL library to applications. This implementation makes use of a Direct Rendering technology, which allows the client to directly write to DMA buffers that are used by the graphics hardware.

Due to the use of direct rendering technology, system designers should take special care to ensure that only trusted clients are allowed to use the OpenGL library. A malicious application could otherwise use direct rendering to destabilize the graphics hardware or, in theory, elevate their permissions on the system.

A system designer can control the access to the direct rendering functionality by limiting the access to the DRI device file located at:

```
/dev/dri/card0
```

The permissions on this device are set by the X server using the information provided in the “DRI” section of the configuration file.

See also:

- “2D Support”
- “3D Support”

B.1 2D Support

Intel® EMGD provides 2D capabilities on Linux* through XRender and OpenVG* 1.1.

B.2 3D Support

Intel® EMGD provides 3D capabilities through several industry-standard APIs, such as OpenGL*, OpenGL* ES, and Direct3D*. These APIs are described in the following sections:

- “OpenGL* APIs”
- “OpenGL* ES 1.1”
- “OpenGL* ES 2.0”
- “EGL”
### B.2.1 OpenGL* APIs

The following OpenGL versions are supported:
- OGLES* 1.1 and 2.0
- OGL* 2.0 and 2.1 (Linux* only)
- EGL*

For general OpenGL* information, visit [http://www.opengl.org/about/overview/](http://www.opengl.org/about/overview/).

- "Inte-Supported OpenGL* APIs for Linux*"
- "Non-Supported Intel® OpenGL APIs"

#### B.2.1.1 Inte-Supported OpenGL* APIs for Linux*

**Table 22. Intel-Supported OpenGL* APIs for Linux* (Sheet 1 of 3)**

| GL_ARB_depth_texture          |                           |
| GL_ARB_draw_buffers           |                           |
| GL_ARB_fragment_program       |                           |
| GL_ARB_fragment_shader        |                           |
| GL_ARB_half_float_pixel       |                           |
| GL_ARB_matrix_palette         |                           |
| GL_ARB_multisample            |                           |
| GL_ARB_multitexture           |                           |
| GL_ARB_occlusion_query        |                           |
| GL_ARB_pixel_buffer_object    |                           |
| GL_ARB_point_parameters       |                           |
| GL_ARB_point_sprite           |                           |
| GL_ARB_shader_objects         |                           |
| GL_ARB_shading_language_100   |                           |
| GL_ARB_shadow                 |                           |
| GL_ARB_shadow_ambient         |                           |
| GL_ARB_texture_border_clamp   |                           |
| GL_ARB_texture_compression    |                           |
| GL_ARB_texture_cube_map       |                           |
| GL_ARB_texture_env_add        |                           |
| GL_ARB_texture_env_combine    |                           |
| GL_ARB_texture_env_crossbar   |                           |
| GL_ARB_texture_env_dot3       |                           |
| GL_ARB_texture_float          |                           |
| GL_ARB_texture_mirror_repeat  |                           |
| GL_ARB_texture_non_power_of_two|                           |
| GL_ARB_texture_rectangle      |                           |
| GL_ARB_transpose_matrix       |                           |

*Note that glinfo will list the supported modes on the current installation.*
Table 22. Intel-Supported OpenGL* APIs for Linux* (Sheet 2 of 3)

| GL_ARB_vertex_blend          |
| GL_ARB_vertex_buffer_object  |
| GL_ARB_vertex_program        |
| GL_ARB_vertex_shader         |
| GL_ARB_window_pos            |
| GL_EXT_abgr                  |
| GL_EXT_bgra                  |
| GL_EXT_blend_color           |
| GL_EXT_blend_equation_separate |
| GL_EXT_blend_func_separate    |
| GL_EXT_blend_minmax          |
| GL_EXT_blend_subtract        |
| GL_EXT_compiled_vertex_array |
| GL_EXT_draw_range_elements   |
| GL_EXT_fog_coord             |
| GL_EXT_framebuffer_object    |
| GL_EXT_multi_draw_arrays     |
| GL_EXT_packed_pixels         |
| GL_EXT_rescale_normal        |
| GL_EXT_secondary_color       |
| GL_EXT_separate_specular_color |
| GL_EXT_shadow_funcs          |
| GL_EXTStencil_two_side      |
| GL_EXTStencil_wrap           |
| GL_EXT_texture_compression_s3tc |
| GL_EXT_texture_cube_map      |
| GL_EXT_texture_edge_clamp    |
| GL_EXT_texture_env_add       |
| GL_EXT_texture_env_combine   |
| GL_EXT_texture_env_dot3      |
| GL_EXT_texture_filter_anisotropic |
| GL_EXT_texture_lod_bias      |
| GL_EXT_texture_object        |
| GL_EXT_texture3D             |
| GL_EXT_vertex_array          |
| GL_IMG_texture_compression_pvrtc |
| GL_NV_blend_square           |
| GL_NV_texgen_reflection      |
| GL_NV_texture_rectangle      |
| GL_S3_s3tc                   |

*Note that glinfo will list the supported modes on the current installation.
Table 22. Intel-Supported OpenGL* APIs for Linux* (Sheet 3 of 3)

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>GL_SGIS_generate_mipmap</td>
</tr>
<tr>
<td>GLX_ARB_get_proc_address*</td>
</tr>
<tr>
<td>GLX_ARB_multisample*</td>
</tr>
<tr>
<td>GLX_EXT_import_context*</td>
</tr>
<tr>
<td>GLX_EXT_texture_from_pixmap</td>
</tr>
<tr>
<td>GLX_EXT_visual_info*</td>
</tr>
<tr>
<td>GLX_EXT_visual_rating*</td>
</tr>
<tr>
<td>GLX_MESA_allocate_memory*</td>
</tr>
<tr>
<td>GLX_MESA_copy_sub_buffer*</td>
</tr>
<tr>
<td>GLX_MESA_swap_control*</td>
</tr>
<tr>
<td>GLX_MESA_swap_frame_usage*</td>
</tr>
<tr>
<td>GLX_OML_swap_method*</td>
</tr>
<tr>
<td>GLX_OML_sync_control*</td>
</tr>
<tr>
<td>GLX_SGI_make_current_read*</td>
</tr>
<tr>
<td>GLX_SGI_swap_control*</td>
</tr>
<tr>
<td>GLX_SGIS_multisample*</td>
</tr>
<tr>
<td>GLX_SGIX_fbconfig*</td>
</tr>
<tr>
<td>GLX_SGIX_pbuffer*</td>
</tr>
<tr>
<td>GLX_SGIX_visual_select_group*</td>
</tr>
</tbody>
</table>

*Note that glinfo will list the supported modes on the current installation.
## B.2.1.2 Non-Supported Intel® OpenGL APIs

### Table 23. Non-Supported OpenGL* APIs

<table>
<thead>
<tr>
<th>API Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GL_3DFX_texture_compression_FXT1</td>
</tr>
<tr>
<td>GL_ARB_color_buffer_float</td>
</tr>
<tr>
<td>GL_ARB_fragment_program_shadow</td>
</tr>
<tr>
<td>GL_ARB_occlusion_query</td>
</tr>
<tr>
<td>GL_ARB_vertex_shader</td>
</tr>
<tr>
<td>GL_EXT_cull_vertex</td>
</tr>
<tr>
<td>GLX_ARB_multisample</td>
</tr>
<tr>
<td>GLX_EXT_import_context</td>
</tr>
<tr>
<td>GLX_EXT_texture_from_pixmap</td>
</tr>
<tr>
<td>GLX_EXT_visual_info</td>
</tr>
<tr>
<td>GLX_EXT_visual_rating</td>
</tr>
<tr>
<td>GLX_MESA_allocate_memory</td>
</tr>
<tr>
<td>GLX_MESA_copy_sub_buffer</td>
</tr>
<tr>
<td>GLX_MESA_swap_control</td>
</tr>
<tr>
<td>GLX_MESA_swap_frame_usage</td>
</tr>
<tr>
<td>GLX_OML_swap_method</td>
</tr>
<tr>
<td>GLX_OML_sync_control</td>
</tr>
<tr>
<td>GLX_SGI_make_current_read</td>
</tr>
<tr>
<td>GLX_SGI_swap_control</td>
</tr>
<tr>
<td>GLX_SGI_swap_control</td>
</tr>
<tr>
<td>GLX_SGI_video_sync</td>
</tr>
<tr>
<td>GLX_SGIS_multisample</td>
</tr>
<tr>
<td>GLX_SGIX_fbconfig</td>
</tr>
<tr>
<td>GLX_SGIX_hyperpipe</td>
</tr>
<tr>
<td>GLX_SGIX_pbuffer</td>
</tr>
<tr>
<td>GLX_SGIX_swap_barrier</td>
</tr>
<tr>
<td>GLX_SGIX_visual_select_group</td>
</tr>
<tr>
<td>WGL_ARB_buffer_region</td>
</tr>
<tr>
<td>WGL_ARB_extensions_string</td>
</tr>
<tr>
<td>WGL_ARB_make_current_read</td>
</tr>
<tr>
<td>WGL_ARB_pbuffer</td>
</tr>
<tr>
<td>WGL_ARB_pixel_format</td>
</tr>
<tr>
<td>WGL_EXT_swap_control</td>
</tr>
</tbody>
</table>
B.2.2  **OpenGL* ES 1.1**

The following OpenGL* ES 1.1 extensions are supported:
- GL_ARB_multisample
- GL_EXT_texture_filter_anisotropic
- GL_EXT_framebuffer_blit
- GL_IMG_texture_compression_pvrtc
- GL_OES_blend_equation_separate
- GL_OES_blend_func_separate
- GL_OES_blend_subtract
- GL_OES_byte_coordinates
- GL_OES_depth24
- GL_OES_depth32
- GL_OES_draw_texture
- GL_OES_element_index_uint
- GL_OES_fbo_render_mipmap
- GL_OES_fixed_point
- GL_OES_framebuffer_object
- GL_OES_mapbuffer
- GL_OES_matrix_get
- GL_OES_point_size_array
- GL_OES_point_sprite
- GL_OES_query_matrix
- GL_OES_read_format
- GL_OES_rgb8_rgba8
- GL_OES_single_precision
- GL_OES_texture_cube_map
- GL_OES_texture_env_crossbar
- GL_OES_texture_mirrored_repeat

B.2.3  **OpenGL* ES 2.0**

The following OpenGL ES 2.0 extensions are supported:
- GL_OES_depth_texture
- GL_OES_standard_derivatives
- GL_OES_texture_3D
- GL_OES_texture_npot
- GL_EXT_texture_type_2_10_10_10_REV
- GL_OES_compressed_paletted_texture
- GL_OES_packed_depth_stencil
**Table 24. Non-Supported OpenGL ES APIs**

<table>
<thead>
<tr>
<th>Non-Supported API Name(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GL_OES_stencil_wrap</td>
</tr>
<tr>
<td>GL_OES_compressed_ETC1_RGB8_texture</td>
</tr>
<tr>
<td>GL_OES_matrix_palette</td>
</tr>
<tr>
<td>GL_OES_EGL_image</td>
</tr>
<tr>
<td>GL_AMD_compressed_3DC_texture</td>
</tr>
<tr>
<td>GL_AMD_compressed_ATC_texture</td>
</tr>
<tr>
<td>GL_OES_texture_float</td>
</tr>
<tr>
<td>GL_OES_texture_half_float</td>
</tr>
<tr>
<td>GL_OES_texture_half_float_linear</td>
</tr>
<tr>
<td>GL_OES_vertex_half_float</td>
</tr>
<tr>
<td>GL_OES_vertex_type_10_10_10_2</td>
</tr>
<tr>
<td>GL_OES_fragment_precision_high</td>
</tr>
</tbody>
</table>

**B.2.4 EGL**

The following EGL functions are supported:
- EGL_NOK_image_shared
- EGL_KHR_image_base
Appendix C Example User_config.c File

/* -*- pse-c -*-
*---------------------------------------------------------------
* Filename: user_config.c
*---------------------------------------------------------------
* Copyright © 2002-2013, Intel Corporation.
*
* This program is free software; you can redistribute it and/or
* modify it under the terms and conditions of the GNU General
* Public License, version 2, as published by the Free Software
* Foundation.
*
* This program is distributed in the hope it will be useful, but
* WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 51 Franklin St - Fifth Floor, Boston, MA
* 02110-1301 USA.
*---------------------------------------------------------------
* Description:
* A file that contains the initial display configuration
* information of the EMGD kernel module. A user can edit this
* file in order to affect the way that the kernel initially
* configures the displays. This file is compiled into the EMGD
* kernel module.
#include "default_config.h"
#include "image_data.h"

/*
 * One array of igd_display_info_t structures should exist for each
 * port that needs to provide a DTD list. Each igd_display_info_t
 * contains the DTD * information for a given resolution/refresh-
 * rate. This is especially needed for analog/VGA ports.
*/
static igd_display_info_t dtd_config1_port3_dtdlist[] = {
};
static igd_param_attr_t attrs_config1_port3[] = {
};
static igd_display_info_t dtd_config1_port2_dtdlist[] = {
};
static igd_param_attr_t attrs_config1_port2[] = {
};

/*
 * Splash screen and splash video functionality is mutually
 * exclusive.
 * If splash_video_data->immediate is enabled then splash screen
 * will not be called. To enable splash screen ensure width and
 * height is non-zero.
*/
static emgd_drm_splash_screen_t splash_screen_data = {
    0x000000, 0,

    0, /* x */
    0, /* y */
    0, /* width */
    0, /* height */
    image_data,
    sizeof(image_data)
};

static emgd_drm_splash_video_t splash_video_data = {
    0,
    /*built_in_test*/
    0,
    /*immediate*/
    0, /*phys_config_dram_start*/
    0, /* igd_pixel_format */
    0, /* src_width */
    0, /* src_height */
    0, /* src_pitch */
}
0, /* dst_x */
0, /* dst_y */
0, /* dst_width */
0, /* dst_height */

};

static igd_param_t config_params[] = {
{
0, 0, 0, 0/* Limit maximum pages usable by GEM (0 = no limit) */
0, 0/* Max frame buffer size (0 = no limit) */
1,/* Preserve registers (should be 1, so VT switches work * and so hat the console will be restored after X server * exits). */
IGD_DISPLAY_DETECT|IGDDISPLAY_MULTI_DVO, /* Display flags */
 /* Display port order, corresponds to the "portorder" module parameter */
 * Valid port types are:
 * IGD_PORT_TYPE_TV
 * IGD_PORT_TYPE_SDVOB
 * IGD_PORT_TYPE_SDVOC
 * IGD_PORT_TYPE_LVDS
 * IGD_PORT_TYPE_ANALOG
 * IGD_PORT_TYPE_DPB
 * IGD_PORT_TYPE_DPC
 */
IGD_PORT_TYPE_SDVOC,IGD_PORT_TYPE_SDVOB,0,0,0,0,0 },
/* Display Params */
/* Port */
3,/* Display port number */
0x100,/* Parameters present */
0x1,/* EDID flag */
0x1,/* Flags when EDID is available */
0x1,/* Flags when EDID is not available */
0,/* DDC GPIO pins */
0,/* DDC speed */
0,/* DDC DAB */
0,/* I2C GPIO pins */
0,/* I2C speed */
0,/* I2C DAB */
/* Flat Panel Info */
0,/* Flat Panel width */
0,/* Flat Panel height */
0,/* Flat Panel power method */
0,/* eDP T3: Delay from LCDVDD to HPD high */
0,/* eDP T7: Delay from valid video source to video on display */
0, /* eDP T9: Delay from backlight disable to end of valid video data */
0, /* eDP T10: Delay from end of valid video data from source to power off */
0/* eDP T12: Power off time */
},
{"DTD Info */
s sizeof(dtd_config1_port3_dtdlist)/sizeof(igd_display_info_t),
dtd_config1_port3_dtdlist
},
{"Attribute Info */
s sizeof(attrs_config1_port3)/sizeof(igd_param_attr_t),
attrs_config1_port3
}
},
{"Port */
2, /* Display port number */
0x100, /* Parameters present */
0x1, /* EDID flag */
0x1, /* Flags when EDID is available */
0x1, /* Flags when EDID is not available */
0, /* DDC GPIO pins */
0, /* DDC speed */
0, /* DDC DAB */
0, /* I2C GPIO pins */
0, /* I2C speed */
0, /* I2C DAB */
{"Flat Panel Info */
0, /* Flat Panel width */
0, /* Flat Panel height */
0, /* Flat Panel power method */
0, /* eDP T3: Delay from LCDVDD to HPD high */
0, /* eDP T7: Delay from valid video source to video on display */
0, /* eDP T9: Delay from backlight disable to end of valid video data */
0, /* eDP T10: Delay from end of valid video data from source to power off */
0/* eDP T12: Power off time */
},
{"DTD Info */
s sizeof(dtd_config1_port2_dtdlist)/sizeof(igd_display_info_t),
dtd_config1_port2_dtdlist
},
{"Attribute Info */
s sizeof(attrs_config1_port2)/sizeof(igd_param_attr_t),
attrs_config1_port2
}
{ /* Port */
  0, /* Display port number */
  0, /* Parameters present */
  0x0, /* EDID flag */
  0x0, /* Flags when EDID is available */
  0x0, /* Flags when EDID is not available */
  0, /* DCC GPIO pins */
  0, /* DCC speed */
  0, /* DCC DAB */
  0, /* I2C GPIO pins */
  0, /* I2C speed */
  0, /* I2C DAB */
  { /* Flat Panel Info */
    0, /* Flat Panel width */
    0, /* Flat Panel height */
    0, /* Flat Panel power method */
    0, /* eDP T3: Delay from LCDVDD to HPD high */
    0, /* eDP T7: Delay from valid video source to video on display */
    0, /* eDP T9: Delay from backlight disable to end of valid video data */
    0, /* eDP T10: Delay from end of valid video data from source to power off */
    0, /* eDP T12: Power off time */
  },
  { /* DTD Info */
    0,
    NULL
  },
  { /* Attribute Info */
    0,
    NULL
  }
},
{ /* Port */
  0, /* Display port number */
  0, /* Parameters present */
  0x0, /* EDID flag */
  0x0, /* Flags when EDID is available */
  0x0, /* Flags when EDID is not available */
  0, /* DCC GPIO pins */
  0, /* DCC speed */
  0, /* DCC DAB */
  0, /* I2C GPIO pins */
  0, /* I2C speed */
  0, /* I2C DAB */
  { /* Flat Panel Info */
    0, /* Flat Panel width */
    0, /* Flat Panel height */
    0, /* Flat Panel power method */
    0, /* eDP T3: Delay from LCDVDD to HPD high */
  },
0,/* eDP T7: Delay from valid video source to video on display */
0,/* eDP T9: Delay from backlight disable to end of valid video data */
0,/* eDP T10: Delay from end of valid video data from source to power off */
0/* eDP T12: Power off time */
},
{"DTD Info"/
0,
NULL}
," Attribute Info */
0,
NULL}
," Port */
0,/* Display port number */
0,/* Parameters present */
0x0,/* EDID flag */
0x0,/* Flags when EDID is available */
0x0,/* Flags when EDID is not available */
0,/* DDC GPIO pins */
0,/* DDC speed */
0,/* DDC DAB */
0,/* I2C GPIO pins */
0,/* I2C speed */
0,/* I2C DAB */
{"Flat Panel Info */
0,/* Flat Panel width */
0,/* Flat Panel height */
0,/* Flat Panel power method */
0,/* eDP T3: Delay from LCDVDD to HPD high */
0,/* eDP T7: Delay from valid video source to video on display */
0,/* eDP T9: Delay from backlight disable to end of valid video data */
0,/* eDP T10: Delay from end of valid video data from source to power off */
0/* eDP T12: Power off time */
},
{"DTD Info"/
0,
NULL}
," Attribute Info */
0,
NULL}
," Port */
0,/* Display port number */
0,/* Parameters present */
0x0,/* EDID flag */
0x0,/* Flags when EDID is available */
0x0,/* Flags when EDID is not available */
0,/* DDC GPIO pins */
0,/* DDC speed */
0,/* DDC DAB */
0,/* I2C GPIO pins */
0,/* I2C speed */
0,/* I2C DAB */
{"Flat Panel Info "/
 0,/* Flat Panel width */
 0,/* Flat Panel height */
 0,/* Flat Panel power method */
 0,/* eDP T3: Delay from LCDVDD to HPD high */
 0,/* eDP T7: Delay from valid video source to video on display */
 0,/* eDP T9: Delay from backlight disable to end of valid video data */
 0,/* eDP T10: Delay from end of valid video data from source to power off */
 0,/* eDP T12: Power off time */
},
{"DTD Info "/
 0,
 NULL
 },
{"Attribute Info "/
 0,
 NULL
 }
{"Port "/
 0,/* Display port number */
 0,/* Parameters present */
 0x0,/* EDID flag */
 0x0,/* Flags when EDID is available */
 0x0,/* Flags when EDID is not available */
 0,/* DDC GPIO pins */
 0,/* DDC speed */
 0,/* DDC DAB */
 0,/* I2C GPIO pins */
 0,/* I2C speed */
 0,/* I2C DAB */
{"Flat Panel Info "/
 0,/* Flat Panel width */
 0,/* Flat Panel height */
 0,/* Flat Panel power method */
 0,/* eDP T3: Delay from LCDVDD to HPD high */
 0,/* eDP T7: Delay from valid video source to video on display */
 0,/* eDP T9: Delay from backlight disable to end of valid video data */
 0,/* eDP T10: Delay from end of valid video data from source to power off */
0x0,/* Quickboot (0x0=default; 0x1=Skip GTT; 0x2=Skip init for ports not in port order list; 0x3=set both 0x1 & 0x2) */
0, /* Quickboot seamless (1 = enabled) */
0, /* Quickboot video input (1 = enabled) */
0, /* Polling (1 = override interrupt support and use polling) */
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
emgd_ovl_config_t initial_ovl_config = {
    {
        {
            /* igd_ovl_gamma_info */
            /* *******************************/
            0x100, /* unsigned int red; */
            0x100, /* unsigned int green; */
            0x100, /* unsigned int blue; */
            IGD_OVL_GAMMA_DISABLE /* unsigned int flags; */
        },
        {
            /* igd_ovl_color_correct_info_t */
            /* *******************************/
            0x8000, /* unsigned short brightness; */
            0x8000, /* unsigned short contrast; */
            0x8000, /* unsigned short saturation; */
            0x8000/* unsigned short hue; */
        }
    },
    {
        {
            /* igd_ovl_gamma_info */
            /* *******************************/
            0x100, /* unsigned int red; */
            0x100, /* unsigned int green; */
            0x100, /* unsigned int blue; */
            IGD_OVL_GAMMA_DISABLE /* unsigned int flags; */
        },
        {
            /* igd_ovl_color_correct_info_t */
            /* *******************************/
            0x8000, /* unsigned short brightness; */
            0x8000, /* unsigned short contrast; */
            0x8000, /* unsigned short saturation; */
            0x8000/* unsigned short hue; */
        }
    },
    {
        {
            /* igd_ovl_gamma_info */
            /* *******************************/
            0x100, /* unsigned int red; */
            0x100, /* unsigned int green; */
            0x100, /* unsigned int blue; */
            IGD_OVL_GAMMA_DISABLE /* unsigned int flags; */
        },
        {
            /* igd_ovl_color_correct_info_t */
            /* *******************************/
            0x8000, /* unsigned short brightness; */
            0x8000, /* unsigned short contrast; */
            0x8000, /* unsigned short saturation; */
            0x8000/* unsigned short hue; */
        }
    }
};
{ /* igd_ovl_color_correct_info_t */
  /*******************************************************************************/
  0x8000,/* unsigned short brightness; */
  0x8000,/* unsigned short contrast; */
  0x8000,/* unsigned short saturation; */
  0x8000/* unsigned short hue; */
}
},
{
  /* igd_ovl_gamma_info          */
  /*******************************************************************************/
  0x100,/* unsigned int red;     */
  0x100,/* unsigned int green;   */
  0x100,/* unsigned int blue;    */
  IGD_OVL_GAMMA_DISABLE /* unsigned int flags;*/
},

/* igd_ovl_pipeblend_t X 1 - known defaults */
{ 
  0x00010204, /* unsigned long zorder_combined*/
  0, /* unsigned long fb_blend_ovl*/
  0, /* unsigned long has_const_alpha*/
  0, /* unsigned long const_alpha*/
  (IGD_OVL_PIPEBLEND_SET_FBBLEND | IGD_OVL_PIPEBLEND_SET_ZORDER |
  IGD_OVL_PIPEBLEND_SET_CONSTALPHA) /*
  unsigned long enable_flags*/
},

}
{
    0x00010204, /* unsigned long zorder_combined*/
    0, /* unsigned long fb_blend_ovl*/
    0, /* unsigned long has_const_alpha*/
    0, /* unsigned long const_alpha*/
    (IGD_OVL_PIPEBLEND_SET_FBBLEND | IGD_OVL_PIPEBLEND_SET_ZORDER |
     IGD_OVL_PIPEBLEND_SET_CONSTALPHA) /*
     unsigned long enable_flags*/
},

);

/**
 * User-configurable parameters for a single CRTC. This configures 
 * the frame buffer settings and the timings for a CRTC. Setting 
 * width and height values to zero will cause the system to auto 
 * detect the best fit values.
 */
emgd_crtc_config_t  crtcs[] = {
    { /* CRTC 0 */
        0, /* frame buffer ID to associate with this crtc */
        1024, /* Display width used to find initial display timings */
        768, /* Display height used to find initial display timings */
        60, /* Display vertical refresh rate to use */
        0, /* X offset */
        0, /* Y offset */
    },

    { /* CRTC 1 */
        0, /* frame buffer ID to associate with this crtc */
        1024, /* Display width used to find initial display timings */
        768, /* Display height used to find initial display timings */
        60, /* Display vertical refresh rate to use */
        0, /* X offset */
        0, /* Y offset */
    },

};
/*  
* The emgd_drm_config_t structure is the main configuration  
* structure for the EMGD kernel module.  
*/

emgd_drm_config_t default_config_drm = {  
    sizeof(crtcs)/sizeof(emgd_crtc_config_t),  
    crtcs,  
    sizeof(initial_fbs)/sizeof(emgd_fb_config_t),  
    initial_fbs,  
    &initial_ovl_config, /* overlay plane initial configuration (can be  
    NULL) */  
    &splash_screen_data,  
    &splash_video_data,  
    sizeof(config_params)/sizeof(igd_param_t),  
    config_params, /* driver parameters from above */  
    0  
};

emgd_drm_config_t *config_drm;