



Intel® Embedded Graphics Drivers and Video BIOS

User's Guide

Version 5.0

February 2006



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. Intel products are not intended for use in medical, life saving, life sustaining, or nuclear facility applications.

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 Intel® Embedded Graphics Drivers and Video BIOS 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.

MPEG is an international standard for video compression/decompression promoted by ISO. Implementations of MPEG CODECs, or MPEG enabled platforms may require licenses from various entities, including Intel Corporation.

This User's Guide as well as the software described in it is furnished under license and may only be used or copied in accordance with the terms of the license. The information in this manual is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Intel Corporation. Intel Corporation assumes no responsibility or liability for any errors or inaccuracies that may appear in this document or any software that may be provided in association with this document.

Except as permitted by such license, no part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without the express written consent of Intel Corporation.

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 ordering number and are referenced in this document, or other Intel literature may be obtained by calling 1-800-548-4725 or by visiting Intel's website at <http://www.intel.com>.

AnyPoint, AppChoice, BoardWatch, BunnyPeople, CablePort, Celeron, Chips, CT Media, Dialogic, DM3, EtherExpress, ETOX, FlashFile, i386, i486, i960, iCOMP, InstantIP, Intel, Intel Centrino, Intel logo, Intel386, Intel486, Intel740, IntelDX2, IntelDX4, IntelSX2, Intel Create & Share, Intel GigaBlade, Intel InBusiness, Intel Inside, Intel Inside logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel Play, Intel Play logo, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel TeamStation, Intel Xeon, Intel XScale, IPLink, Itanium, MCS, MMX, MMX logo, Optimizer logo, OverDrive, Paragon, PC Dads, PC Parents, PDCharm, Pentium, Pentium II Xeon, Pentium III Xeon, Performance at Your Command, RemoteExpress, SmartDie, Solutions960, Sound Mark, StorageExpress, The Computer Inside., The Journey Inside, TokenExpress, VoiceBrick, VTune, and Xircom are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

Copyright © 2006, Intel Corporation. All rights reserved.

Contents

1	Introduction	11
1.1	Purpose	12
1.2	Intended Audience	12
1.3	Related Documents	12
1.4	Conventions	13
1.5	New Features for Version 5.0	13
1.6	Acronyms and Terminology	14
1.7	Downloading the IEGD and Video BIOS	16
1.8	Configuring and Installing the IEGD and Video BIOS	17
2	Architectural Overview	19
2.1	Introduction	19
2.1.1	Display Options	21
2.1.1.1	Types of Displays	21
2.1.1.2	Display Configuration	21
2.2	Features	22
2.2.1	Chipsets Supported	22
2.2.2	OS and API Support	23
2.2.3	EDID-Less Configuration	23
2.2.3.1	EDID-Less Panel Type Detection	23
2.2.4	DVO Devices	24
2.2.5	Rotation	24
3	Platform Configuration	27
3.1	Configuration Overview	27
3.2	System BIOS Settings	27
3.2.1	GMCH PCI Device Enabling	27
3.2.2	Graphics Mode Select (GMS)	27
3.2.3	AGP (Accelerated Graphics Port) Aperture Size	28
3.3	VBIOS and Driver Configuration	28
3.4	pcf2iegd Usage	29
3.4.1	pcf2iegd -ver	29
3.4.2	pcf2iedg -h	29
3.4.3	pcf2iegd -asm	29
3.4.4	pcf2iegd -cfg	30
3.4.5	pcf2iegd -inf	30
3.4.6	pcf2iegd -x	30
3.4.7	pcf2iegd -bin	31
3.5	Parameter Configuration Format	31
3.6	Sample PCF File	42
3.7	Display Detection and Initialization	47
3.7.1	Display Detect Operation	47
3.7.2	Detectable Displays	49
3.8	Advanced EDID Configuration	50
3.8.1	Sample Advanced EDID Configurations	50
3.9	External PCI Graphics Adaptor as Primary Device	51

3.10	Enhanced Clone Mode Support.....	54
3.10.1	Sample Extended Clone Mode PCF File	55
4	VBIOS.....	59
4.1	Overview.....	59
4.2	System Requirements	59
4.3	Building the VBIOS with the User Build System (UBS)	59
4.3.1	Unzipping the VBIOS Zip File	60
4.3.2	Prerequisites.....	60
4.3.3	Selecting the Build Folder.....	60
4.3.4	Configuring the PCF File for the Video BIOS	61
4.3.4.1	COMMON_TO_PORT	61
4.3.4.2	post_display_msg	62
4.3.4.3	OEM Vendor Strings.....	62
4.3.4.4	Default Mode Settings.....	62
4.3.4.5	Default Refresh Settings	63
4.3.4.6	default_vga_height.....	63
4.3.5	Building the VBIOS	63
4.4	VBIOS and Driver Compatibility.....	63
4.4.1	Data Dependencies Between VBIOS and Intel Graphics Drivers.....	63
4.5	VESA and VGA Video Modes.....	64
5	Legacy VBIOS.....	69
5.1	Overview.....	69
5.1.1	VBIOS Model	70
5.2	Panel Detection	71
5.3	Configuration Using User Build System (UBS).....	72
5.3.1	Overview.....	72
5.3.2	Requirements	72
5.3.3	VBIOS Launcher.....	73
5.3.4	VBIOS Configuration File.....	73
5.3.4.1	Version Selection Category	73
5.3.4.2	Build Selection Category.....	73
5.3.4.3	General Selection Category.....	75
5.3.4.4	PORT_CONFIG Selection Category.....	76
5.3.4.5	PORT_DEVICE Selection Category	78
5.3.4.6	BOOT Display Attachment Selection Category	81
5.3.5	VBIOS Customization Tool (VCT)	81
5.3.6	VBIOS Tips.....	82
5.4	System BIOS Interface	82
5.5	VBIOS and Driver Compatibility.....	82
5.5.1	Data Dependencies Between VBIOS and Intel Graphics Drivers.....	82
5.6	Video Modes.....	82
6	Configuring and Installing Microsoft Windows* Drivers	85
6.1	Overview.....	85
6.2	Configuration Information	85
6.2.1	Universal INF Configuration.....	85
6.2.2	INF File Backward Compatibility.....	85
6.2.2.1	INF File Backward Compatibility with IEGD Version 4.0.....	86
6.2.3	Dual Panel Configuration.....	86

6.2.4	Intel® 855GME Dual Display Example	86
6.2.5	Creating Registry Settings for Graphics Driver INF file.....	87
6.2.6	Dynamic Port Driver Configuration	89
6.2.6.1	DestinationDirs.....	89
6.2.6.2	SourceDisksFiles	90
6.2.6.3	Name and File Association	90
6.2.6.4	PortDrivers Registry Key.....	91
6.2.7	Creating an .sld file for Microsoft Windows XP Embedded* Systems	91
6.2.8	Changing Default Display Mode	92
6.3	Installing the IEGD on Microsoft Windows*	92
6.4	Uninstalling Version 5.0 of the Driver	94
6.5	Run-Time Operation	95
6.6	Viewing and Changing the Driver Configuration From Microsoft Windows*	95
7	Installing and Configuring Microsoft Windows CE* Drivers	101
7.1	Overview	101
7.2	Microsoft Windows CE* .NET Installation	101
7.2.1	Prerequisites	101
7.2.2	Integrating IEGD With Microsoft Windows CE* .NET Platform Builder.....	102
7.3	Microsoft Windows CE* .NET Configuration.....	103
7.3.1	Mode Setting Through the Registry	103
7.3.2	Port Driver-Specific Installation Options	107
7.3.3	Microsoft Windows CE* .NET Memory Model -- Reserved Memory.....	108
7.3.4	Determining Graphics Memory Size	109
7.3.5	Framebuffer and Video Surface Size.....	109
7.3.6	Video Surface Allocation Rule	110
7.3.7	Driver General Configuration Selection	110
7.3.8	Display Mode Configuration.....	111
7.3.8.1	Vertical Extended Display Mode	112
7.3.9	Per Port Platform Customization.....	112
7.3.9.1	Per Port Customization -- Custom DTD Mode Timings	113
7.3.9.2	Per Port Customization -- Custom DVO GPIO Pin Settings	114
7.3.9.3	Per Port Customization -- Custom Flat Panel Controls.....	114
7.3.9.4	Per Port Customization -- Attribute Initialization	114
7.3.10	System to Video Stretch Blit	115
7.3.11	Blend Filtering	115
7.3.12	Catalogue Feature File	116
7.3.13	Text Anti-Aliasing	116
7.3.14	Dynamic Loading of Port Drivers	116
7.3.15	NOTEARING Support for Smooth Blits.....	116
7.3.16	video.reg File Backward Compatibility	117
8	Linux Installation and Configuration	125
8.1	Overview	125
8.2	Prerequisites	125
8.3	Installation.....	126
8.4	Configuration	130
8.4.1	Supported Hardware.....	130
8.4.2	Configuration Overview	130
8.4.2.1	Linux Configuration Using the pcf2iegd Utility	130
8.4.2.2	Editing the Linux Configuration Directly	131

8.4.3	The Linux Configuration File.....	134
8.4.3.1	Device Section.....	134
8.4.3.2	Screen Section.....	134
8.4.3.3	Monitor Section.....	135
8.4.3.4	ServerLayout Section.....	135
8.4.3.5	ServerFlags Section.....	135
8.4.4	XFree86 and Xorg Configuration Options.....	136
8.4.5	Sample Dual Independent Head (DIH) Configuration.....	138
8.4.6	Video Memory Management.....	139
8.4.7	Graphics Port Initialization.....	139
8.4.8	EDID-Less Bin Configuration File.....	140
8.4.8.1	EDID File Format.....	140
8.4.9	Sample Advanced EDID Configurations for Linux OS.....	142
8.5	Run Time Operation.....	143
8.5.1	Runtime Configuration GUI (IEGDGUI).....	143
8.5.1.1	IEGDGUI Setup.....	143
8.5.1.2	Using the IEGDGUI Runtime Configuration Utility.....	144
A	PCF File Syntax.....	149
B	Example INF File.....	155
C	Port Driver Attributes.....	163
C.1	Standard Port Driver Attributes.....	163
C.2	Port Driver Customized Attributes.....	164
C.2.1	Chrontel* CH7009/CH7010 Port Driver TV Attributes.....	165
C.2.2	Chrontel* CH7017/CH7305 Port Driver Attributes.....	168
C.2.3	Chrontel* CH7307 Port Driver Attributes.....	173
C.2.4	Chrontel* CH7308 Port Driver Attributes.....	174
C.2.5	Focus* FS453/FS454 Port Driver TV Attributes.....	175
C.2.6	National Semiconductor* NS387R Port Driver LVDS Attributes.....	177
C.2.7	Silicon Image* Sil164 Port Driver DVI Attributes.....	178
C.2.8	Silicon Image* Sil 1362/Sil 1364Port Driver DVI Attributes.....	179
C.3	Chipset and Port Driver-Specific Installation Information.....	180
D	Intel 5F Extended Interface Functions.....	183
	Index.....	185

Figures

1	Intel Embedded Graphics Suite.....	19
2	Graphics Driver Architecture.....	20
3	Firmware Architecture.....	20
4	Conversion of PCF for VBIOS and Driver Configuration.....	29
5	Hierarchy of PCF File Block Headings.....	32
6	External PCI Graphics Card as Primary Driver and IEGD as Secondary Driver.....	52
7	IEGD as Primary Driver and External PCI Graphics Card as Secondary Driver.....	53
8	IEGD as Primary Driver With Two Displays and External PCI Driving a Tertiary Display.....	54
9	Extended Clone Mode Displays.....	55
10	Video BIOS Directory Structure.....	61
11	VBIOS Model.....	71
12	Build Settings.....	73

13	General Options.....	75
14	Port Configuration Options	76
15	Port Device Options.....	79
16	Boot Options	81
17	Runtime Configuration GUI -- Driver Info Tab	96
18	Runtime Configuration GUI -- Display Config Tab.....	97
19	Runtime Configuration GUI -- Display Attributes Tab	98
20	Runtime Configuration GUI -- Mode Tab Screen.....	99
21	Sample FILES Block from platform.bib File	103
22	Typical Memory Map Using Static Memory Model	108
23	Sample video.reg File	118
24	Sample XF86Conf File.....	132
25	Sample DIH Configuration	138
26	Linux Runtime Configuration GUI-- Driver Info Tab.....	145
27	Linux Runtime Configuration GUI -- Display Config Tab	146
28	Linux Runtime Configuration GUI -- Display Attributes Tab	147
29	Linux Runtime Configuration GUI -- Mode Tab	148

Tables

1	Acronyms and Terminology	14
2	Types of Displays	21
3	Display Configuration Definitions.....	21
4	Supported Display Configurations	22
5	Chipsets Supported by the Intel Embedded Graphics Suite.....	22
6	DVO/SDVO Devices Supported	24
7	GMCH Device 2, Function 1 BIOS Setting	27
8	GMS Settings.....	28
9	Parameter Configuration Format	33
10	Pin Pair Definitions	41
11	Detectable Displays	49
12	Sample Advanced EDID Configurations.....	50
13	Supported VGA Video Display Modes.....	64
14	VESA Modes Supported by Video BIOS	65
15	Example Panel ID Definitions	71
16	Device IDs	79
17	Standard VGA Video Display Modes.....	83
18	VESA Modes Supported by Legacy VBIOS	84
19	Example of Intel® 855GME Dual Display Parameter Setting	86
20	Registry Keys.....	104
21	Supported Driver Options	136
22	Sample Advanced EDID Configurations for Linux OS.....	142
23	Standard Port Driver Attributes.....	163
24	Chrontel* CH7009/CH7010 Port Driver TV Attributes	165
25	Chrontel* CH7017/CH7305 Port Driver Attributes	168
26	Chrontel* CH7017/CH7307 Port Driver Attributes	173
27	Chrontel* CH7308 Port Driver Attributes	174
28	Focus* FS453/FS454 Port Driver TV Attributes	175
29	National Semiconductor* NS387R Port Driver LVDS Attributes	177
30	Silicon Image* Sil164 Port Driver DVI Attributes	178

31	Silicon Image* Sil 1362/Sil 1364 Port Driver DVI Attributes	179
32	Default DVO Search Order	180
33	Default GPIO Pin Pair Assignments	180
34	Default IC2 Device Address Byte Assignment	180
35	Summary of Intel 5F Extended Interface Functions	183

Revision History

Date	Revision	Description
February 2006	008	Updated for use with Version 5.0 of the product, including support for the Intel® 852GM, Intel® 945G, and Intel® 945GM chipsets, the Silicon Image* Sil 1362 and Sil 1364 SDVO transmitters, and External PCI as a Primary graphics adaptor.
October 2005	007	Updated for use with Version 4.1 of the product.
June 2005	006	Updated for use with Version 4.0 of the product, including support for the Intel® 915GV and Intel® 915GM chipsets, the Chronitel* CH7307 and Chronitel* CH7308 SDVO transmitters, and Advanced EDID Configuration.
May 2005	005	Updated for use with Version 3.4 of the product, including use of the enhanced Video BIOS, Windows* installer/uninstaller, runtime configuration GUIs, and display discovery feature.
July 2004	004	Updated for use with Version 3.2 of the product, including use of the dynamic port driver feature.
May 2004	003	Updated for usage with version 3.1 of the product, including details on PCF format and usage, Universal INF format, and updates to the User Build System.
February 2004	002	Updated chipset support to reflect current Embedded IA32 roadmap.
February 2004	001	Initial Release

Introduction

1

The Intel® Embedded Graphics Drivers (IEGD) comprise a suite of multi-platform graphics drivers designed to meet the requirements of embedded applications. The drivers run on Embedded Intel® Architecture (eIA) chipsets, including the following:

- Intel® 815/815E
- Intel® 845GV
- Intel® 852GM
- Intel® 852GME
- Intel® 855GME
- Intel® 915GV
- Intel® 915GM
- Intel® 945G
- Intel® 945GM

The IEGD supports four types of display devices:

- Analog CRT
- LVDS flat panels
- TMDS DVI displays
- TV Output

The IEGD is designed to work with fixed-function platforms, such as Point-of-Sale (POS) devices, ATM machines, gaming devices, etc. It can be configured to work with various hardware and software platforms and supports both Microsoft Windows* and Linux operating systems, including embedded versions of these operating systems.

The Intel Embedded Graphics Suite consists of both the IEGD and a Video BIOS (VBIOS) component. These two components are configurable and work together to provide a wide range of features. This document provides information on configuring and using both the IEGD and the VBIOS.

The IEGD provides the following features:

- Enhanced VBIOS support
- Dynamic Port Drivers
- Support for Dual Independent Head (DIH) displays
- Support of a Universal INF file
- EDID and EDID-less display support
- Display discovery and initialization
- Direct 3D* support
- Installer/Uninstaller GUI for Microsoft Windows*

- Runtime configuration GUI for Microsoft Windows* and Linux

1.1 Purpose

This manual provides information on both firmware and software, providing hardware design considerations, installation requirements, and static configuration options.

1.2 Intended Audience

This document is targeted at 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, as well as end users.

1.3 Related Documents

The following documents provide additional information on the hardware supported by the IEGD.

- *Mobile Intel® 915PM/GM/GMS and 910GML Express Chipset Datasheet*
(Document Number: 305264-001)
- *Intel® 915G/915GV/915P Express Chipset Datasheet*
(Document Number: 304467-005)
- *Intel® I/O Controller Hub 6 (ICH6) Family Datasheet*
(Document Number: 301473-002)
- *Intel® 815 Chipset Family: 82815 Graphics and Memory Controller Hub (GMCH) Datasheet*
(Order Number: 298351-002)
- *Intel® 845G/845GL/845GV Chipset Datasheet: Intel® 82845G/82845GL/82845GV Graphics and Memory Controller Hub (GMCH)*
(Order Number: 290746-002)
- *Intel® 852GME Chipset GMCH and Intel® 852PM Chipset MCH Datasheet*
(Order Number: 253027-004)
- *Intel® 855GM/GME Chipset Graphics and Memory Controller Hub (GMCH) Datasheet*
(Order Number: 252615-005)
- *Intel® Embedded Graphics Drivers for Embedded Intel® Architecture-based Chipsets Product Brief*
- *Intel® 815 Chipset: Graphics Controller Programmer's Reference Manual (PRM)*
(Order Number: 298237-001)
- *Integrated Dual Independent Display on Intel® Digital Security Surveillance Multifunction Platforms Application Brief*
- *Display Panel Debugging with the Intel Graphics Memory Controller Hub*
(Order Number: 305964-001)
- *VESA BIOS Extensions/Display Data Channel Standard*, available at the following website:

<http://www.vesa.org/public/VBE/VBEDDC11.PDF>

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*, available at the following website:

<http://www.vesa.org/public/VBE/vbe3.pdf>

Contains information on the VESA BIOS Extension (VBE) specification for standard software access to graphics display controllers that support resolutions, color depths, and frame buffer organizations beyond the VGA hardware standard.

1.4 Conventions

The following conventions are used throughout this document.

Boldface	Represents text that you type and text that appears on a screen.
<i>Italics</i>	Introduces new terms and titles of documents.
Courier New	Identifies the names of files, executable program names, and text that appears in a file.
Angle Brackets (<>)	Enclose variable values in syntax or value ranges that you must replace with actual values.
Vertical Bar ()	Used to separate choices (for example, TRUE FALSE)

1.5 New Features for Version 5.0

The IEGD 5.0 release provides the following features and capabilities:

- Intel® 945G Chipset support (2D support only)
- Intel® 945GM Chipset support (2D support only)
- Intel® 852GM Chipset support
- Dual Digital Transmitter support on Serial DVO
- Support for Silicon Image* SiL1362/SiL1364 SDVO transmitters
- Enhanced Clone mode support for use with different sized displays
- Ability to use PCI based graphics as the primary display and integrated graphics as a secondary display
- Ability to configure port names used in the Runtime GUI
- Ability to change resolution and refresh rates for Clone displays in the Runtime GUI
- Expanded 2D acceleration in MS Windows CE .NET* (4.2 and 5.0)
- Support for SUSE Enterprise Server (SLES) 9

- Windows XP SP2 support
- Improved method for generating Linux configurations using the new `-x` option of the `pcf2iegd` utility.
- A new `int15` configuration parameter for enabling or disabling 5F System BIOS 15h interrupt hooks.

This release also contains fixes for errata. Refer to the Errata document at the following website for details.

<http://www.intel.com/design/intarch/specupdt/309380.htm>

1.6 Acronyms and Terminology

Table 1 lists the acronyms and terminology used throughout this document.

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

Term	Description
ADD Card	APG Digital Display. An adapter card that can be inserted into the AGP port of Intel chipset family-based systems. ADD cards allow configurations for TV-out, LVDS, and TMDS output (i.e., televisions, digital displays, and flat panel displays).
AGP	Accelerated Graphics Port. An interface specification that allows 3-D graphics to be displayed smoothly and quickly on a display device.
AIM	Add In Module.
BDA	BIOS Data Area. A storage area that contains information about the current state of a display, including mode number, number of columns, cursor position, etc.
BIOS	Basic Input/Output System. The IEGD interacts with two BIOS systems: system BIOS and Video BIOS (VBIOS). VBIOS is a component of the system BIOS.
Clone Display Configuration	A type of display configuration that drives two display devices, each displaying the same content, but can have different resolutions and (independent) timings. Compare Twin Display Configuration and DIH Display Configuration.
DirectDraw*	A component of the DirectX* Graphics API in Microsoft Windows*.
DIH Display Configuration	Dual Independent Head. A type of display configuration that supports two displays with different content on each display device. The IEGD supports Extended Mode for Microsoft Windows* systems and Xinerama for Linux systems.
DTD	Detailed Timing Descriptor. A set of timing values used for EDID-less devices.
DVI	Digital Video Interface.
DVO	Digital Video Out. A port on the GMCH that allows connection to a digital transmitter, either an ADD card or on-board transmitter, and permits connections to various digital devices, such as TVs, LVDS flat panel displays, and TMDS devices. The GMCH provides up to three DVO ports named DVOA, DVOB, and DVOC.

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

Term	Description
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 IEGD. IEGD 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.
eIA	Embedded Intel® Architecture.
EMI	Electromagnetic Interference.
Extended Clone Mode	A feature that allows you to have different sized displays in clone mode.
Frame Buffer	A region of physical memory used to store and render graphics to a display.
GDI	Graphics Device Interface. A low-level API used with Microsoft Windows* operating systems.
GMCH	Graphics and Memory Controller Hub.
GMS	Graphics Mode Select (stolen memory).
HAL	Hardware Abstraction Layer. An API that allows access to the Intel® 815/815E, 845GV, 855GME, and 852GME chipsets.
IAL	Interface Abstraction Layer. An API that allows access to graphics interfaces including the GDI, DirectDraw*, XFree86.
IEGD	Intel® Embedded Graphics Drivers
INF file	A standard Microsoft Windows* text file, referred to as an information file, used by Microsoft Windows* to provide information to the driver. The default .inf file for the IEGD is <code>ssigd.inf</code> . You can create customized parameters in the .pcf file and take the output from the <code>pcf2iegd -inf</code> program and add it to the .inf file.
LVDS	Low Voltage Differential Signaling. Used with flat panel displays, such as a laptop computer display.
Linux/XFree86	Open Source for XWindows* used on Linux systems.
NTSC	National Television Standards Committee. A TV standard used in North and Central America and in Japan.
OAL	Operating System Abstraction Layer. An API that provides access to operating systems, including Microsoft Windows* and Linux.
Option ROM	Code which 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.
PAL	Phase Alternating Lines. A TV standard used in Europe, South America, Africa, and Australia.
PCI	Peripheral Component Interface.
Port Driver	A driver used with the DVO interfaces of the Graphics and Memory Controller Hub (GMCH).
POST	Power On Self Test.
RAL	Resource Abstraction Layer. An API that serves as a bridge between the HAL and other system resources, such as memory, frame buffers, etc.

Table 1. Acronyms and Terminology (Continued) (Sheet 3 of 3)

Term	Description
Reserved Memory	A region of physical memory in a Windows* CE system set aside for BIOS, VBIOS, and Graphics Driver operations. Reserved memory can be configured to be used by the operating system and other applications when not in use by the BIOS.
sDVO	Serial Digital Video Output.
Single Display Configuration	A type of display configuration that supports one and only one display device.
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.
TMDS	Transitioned Minimized Differential Signaling. Used with DVI displays, such as plasma TVs.
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.
Twin Display Configuration	A type of display configuration that supports two display devices each of which has the same content, resolution, and timings. Compare Clone Display Configuration.
UBS	User Build System. A process for building a VBIOS.
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.

1.7 Downloading the IEGD and Video BIOS

Both the IEGD and the Video BIOS (VBIOS) are available from the drivers Download Finder on www.intel.com or can be directly downloaded from the following website:

http://www.intel.com/design/intarch/swsup/graphics_drivers.htm

The following files are available from the Download Finder:

- Intel® Embedded Graphics Driver Video BIOS - v5.0 (winzip file)
- Intel® Embedded Graphics Driver Legacy Video BIOS - v3.2.1 (winzip file)
- Intel® i81x Embedded Video BIOS - v1.3 (winzip file)
- Intel® Embedded Graphics Driver for Microsoft Windows* 2000, Windows XP* Professional, Windows* XP Embedded, and Windows NT* - v5.0 (winzip file)
- Intel® Embedded Graphics Driver for Microsoft Windows CE* .NET - v4.2 (winzip file)
- Intel® Embedded Graphics Driver for Microsoft Windows CE* .NET - v5.0 (winzip file)

- Intel® Embedded Graphics Driver for Linux - v5.0 (tgz file)

Note: The Embedded Video BIOS version 5.0 is recommended for use with each of the graphics drivers in most cases. The Legacy Video BIOS version 3.2.1 may be used on Intel 845G, 852GME, and 855GME Chipset based platforms. On platforms using the Intel 815 or 815E Chipset, you may use the VBIOS provided with the platform, or the i81x Embedded Video BIOS version 1.3. Click the following link to see the FAQ page for details on the differences of these versions.

http://www.intel.com/design/intarch/swsup/graphics_faq.htm

A typical download consists of downloading two compressed files: one for the IEGD for your system and one for the VBIOS. It is highly recommended that you unzip the files for the VBIOS in a folder near the root of the filesystem.

Once you have downloaded the driver and VBIOS, you can configure and customize them following the procedures in this document. Once they have been configured, you can integrate the VBIOS with the system BIOS ROM and install the IEGD on your operating system.

1.8 Configuring and Installing the IEGD and Video BIOS

The following procedure outlines the process of obtaining, configuring, and installing the IEGD and the VBIOS.

1. Download the VBIOS and the IEGD software from the following website:

http://www.intel.com/design/intarch/swsup/graphics_drivers.htm

2. Unzip the files. For the Video BIOS files, unzip them to a folder near the top of the filesystem root as there is character limitation in DOS. Also note that the User Build System configuration tools require relative locations of the subdirectories. Once you have unzipped the files, do not move or rename any of the folders or files.
3. Configure the VBIOS. If you are using the legacy VBIOS, update the VBIOS Configuration File (`def_eg.txt`). Refer to [Chapter 5, “Legacy VBIOS”](#) for instructions on configuring the legacy VBIOS.
If you are using the new VBIOS, use the Parameter Configuration File (PCF). A `sample.pcf` file is included in the download. Refer to [Section 3.5, “Parameter Configuration Format”](#) on page 31 in [Chapter 3, “Platform Configuration”](#) for information pertaining to the PCF file.
4. Configure the System BIOS. For instructions, see [Section 3.2, “System BIOS Settings”](#) on page 27.
5. Build the VBIOS Option ROM using the User Build System process described in [Chapter 4, “VBIOS”](#). If you are using the legacy VBIOS, refer to [Chapter 5, “Legacy VBIOS”](#).
6. Integrate the VBIOS Option ROM with the platform’s firmware image using tools provided by your system BIOS vendor. For example, if you are using the American Megatrends Inc.* (AMI) system BIOS, use `mmttools.exe`.
7. If required, configure the Driver by updating the Parameter Configuration File. Refer to [Chapter 3, “Platform Configuration”](#).
8. Install the IEGD on your operating system. If you are installing to a Microsoft Windows* system, use the Microsoft Windows* installer program described in [Section 6.6, “Viewing and Changing the Driver Configuration From Microsoft Windows*”](#) on page 95.

If you are installing to a Linux operating system, follow the installation procedure described in [Chapter 8, “Linux Installation and Configuration”](#).

Once the IEGD is installed, you can make runtime configuration changes by using the Microsoft Windows* or Linux runtime configuration GUI. See [Section 6.6, “Viewing and Changing the Driver Configuration From Microsoft Windows*”](#) on page 95 for information on using the Microsoft Windows* Runtime Configuration GUI and [Section 8.5.1, “Runtime Configuration GUI \(IEGDGUI\)”](#) on page 143 for using the Linux Configuration GUI. Changes you make to the IEGD during runtime take effect immediately and there is no need to reboot the operating system.

Architectural Overview

2

2.1 Introduction

The Intel Embedded Graphics Suite (IEGS) is composed of a runtime graphics driver and a Video BIOS (VBIOS) firmware component. (See [Figure 1](#) through [Figure 3](#).) Both the driver and VBIOS control the GMCH to perform display and render operations. The VBIOS 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 Microsoft Windows* operating systems.

Figure 1. Intel Embedded Graphics Suite

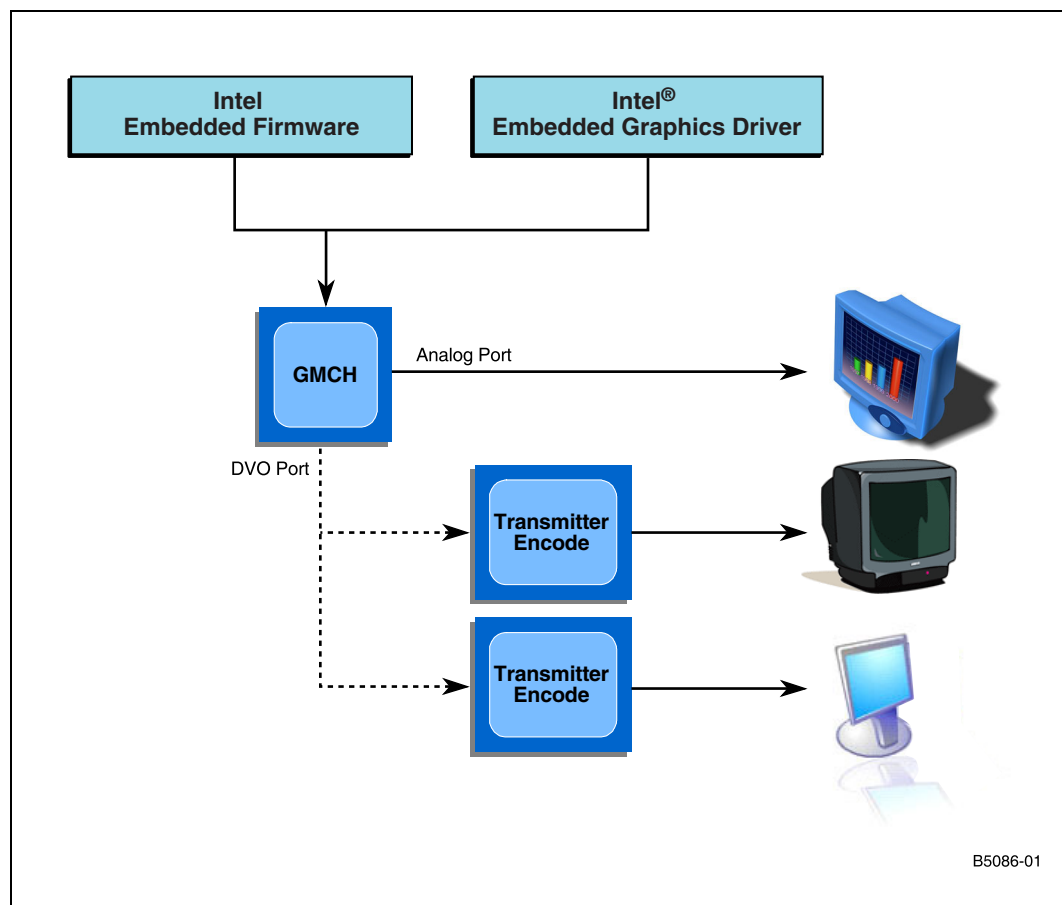


Figure 2. Graphics Driver Architecture

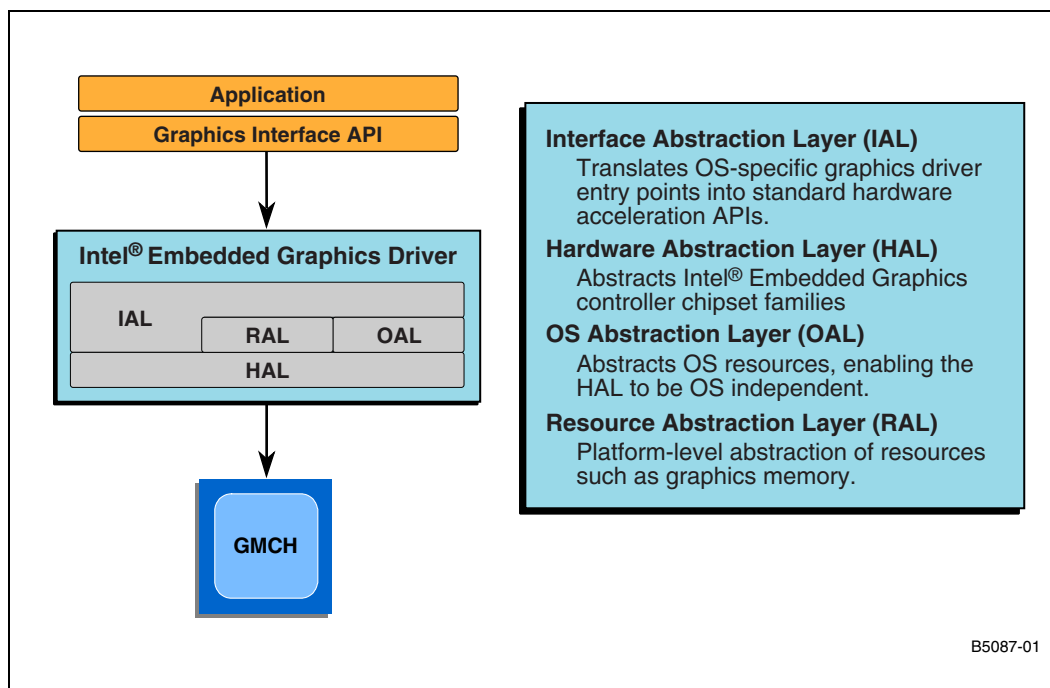
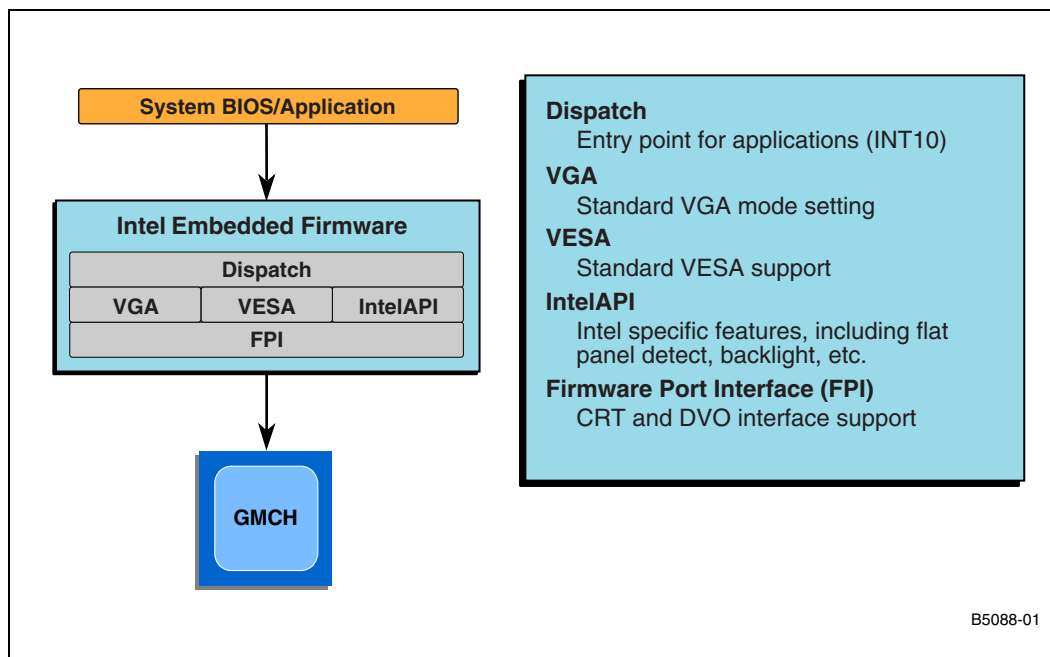


Figure 3. Firmware Architecture



2.1.1 Display Options

The following section describes the types of displays and configurations supported by the Intel Embedded Graphics Driver.

2.1.1.1 Types of Displays

The table below lists the types of displays supported by the IEGD.

Table 2. Types of Displays

Display	Description
CRT	Analog CRT, natively supported with RGB signaling.
Flat Panel	TMDS and LVDS compliant flat panels are supported with the use of an external transmitter via a DVO or sDVO port. Integrated LVDS flat panels are also natively supported on the Intel® 852GME, Intel® 855GM, and Intel® 915GM chipsets.
TV	TV-out is supported with the use of external encoder via a DVO or sDVO port. NOTE: The integrated TV Out signal of the Intel® 915GM Chipset is not currently supported in IEGD.

2.1.1.2 Display Configuration

IEGD supports driving two displays simultaneously. Several configurations are supported, dependent on operating system and chipset. The various display configuration are described in the table below.

Table 3. Display Configuration Definitions

Display Configuration Mode	Description
Single	Normal desktop configuration, single monitor
Twin	Two displays, same content, single resolution, same timings
Clone	Two displays, same content, different resolutions, independent timings
DIH	Dual Independent Head. Two displays, different content, independent resolutions

Table 4 summarizes which display configurations are supported by Intel chipsets.

Table 4. Supported Display Configurations

Chipset	Operating System			
	Windows* 2000	Windows XP*	Windows CE* .NET	Linux
Intel® 815/815E, Intel® 845GV	Single Twin	Single Twin	Single Twin	Single Twin
Intel® 852GME, Intel® 852GM, Intel® 855GME	Single Twin Clone DIH	Single Twin Clone DIH	Single Twin Clone	Single Twin Clone DIH
Intel® 915GV, Intel® 945G	Single Twin Clone	Single Twin Clone	Single Twin Clone	Single Twin Clone
Intel® 915GM, Intel® 945GM	Single Twin Clone DIH	Single Twin Clone DIH	Single Twin Clone	Single Twin Clone DIH

Twin and Clone modes are supported by IEGD through custom APIs. In contrast, DIH is supported natively by both Microsoft Windows* and Linux (XFree86 and X.org).

2.2 Features

The following sections describes major features supported by IEGD.

2.2.1 Chipsets Supported

The following table details the chipsets that the IEGS supports.

Table 5. Chipsets Supported by the Intel Embedded Graphics Suite

Chipset	IEGD Legacy VBIOS Support	IEGD VBIOS Support	IEGD Support
Intel 815/815E	No	No	Yes
Intel 845GV	Yes	Yes	Yes
Intel 852GME Intel 852GM	Yes No	Yes Yes	Yes Yes
Intel 855GME	Yes	Yes	Yes
Intel 915GV Intel 915GM	No No	Yes Yes	Yes Yes
Intel 945G Intel 945GM	No No	Yes Yes	Yes Yes

All supported chipsets provide support for a single analog output for CRTs. In addition, digital monitors, flat panels and TVs are supported through the GMCH DVO and sDVO interface.

Note: The 815/815E chipsets are not supported by either the legacy IEGD VBIOS nor the new 5.0 VBIOS. These chipsets work only with VBIOS provided by the system BIOS manufacturer or the i81x Embedded Video BIOS.

2.2.2 OS and API Support

The IEGD and Video BIOS support the following OS and APIs:

- Linux XFree86 and X.org
- Microsoft Windows* 2000, Windows XP*, Windows XP Embedded*
 - DirectX* 8.1 (DirectDraw* and Direct3D*)
 - DirectX* 9 (DirectDraw* and Direct3D*)
- Microsoft Windows CE* .NET* 4.2 (DirectDraw* and Direct3D*)
- Microsoft Windows CE* .NET* 5.0 (DirectDraw* only)
- Microsoft Windows NT* 4.0 with DirectX* 3 (DirectDraw* only)

2.2.3 EDID-Less Configuration

EDID-less support is the ability to run a display panel that does not have display timing information within the panel. Therefore, the user has to provide the display timing information to the graphics drivers. For the IEGD, this must be done through:

- VBIOS User Build System for the VBIOS
- Configuration file for the graphics drivers.

This document describes only the necessary edits to the configuration files that are required to implement the graphics driver and VBIOS, and not specific settings for EDID-less panel configuration. Please refer to the manufacturer's specifications for the DTD settings to use for your EDID-less panels.

2.2.3.1 EDID-Less Panel Type Detection

The Intel Embedded Graphics Suite supports EDID-less displays that do not export timing modes. This is accomplished by allowing configuration of a Detailed Timing Descriptor (DTD), and associating that DTD with a specific display port. The IEGS provides further flexibility in allowing numerous DTDs to be defined and having the selection of the DTD be configurable through selection of Configuration IDs. The selection of the Configuration ID can be done from the System BIOS, as long as it supports the Intel 5F40h function and passes the appropriate Configuration ID to the VBIOS. The VBIOS in turn notifies the Graphics Driver of which Configuration ID is active. This is not required however, but the VBIOS and/or Graphics Driver require the Configuration ID to be set prior to installation.

2.2.4 DVO Devices

The IEGD supports many third-party digital transmitters connected to the DVO ports of the GMCH. The driver code that supports each of these devices is abstracted and is a separate driver called a port driver. Port drivers can be dynamically loaded at the time IEGD is initialized, and IEGD can be configured to allow any number of these port drivers to be loaded. By default, all the port drivers for the devices listed in the following table as Included in Release Package will be loaded by default if the corresponding transmitter is detected. If a port driver is not specified in the configuration before installation, that device will not be detected, and the port driver will not be loaded. The configuration can be modified before installation to prevent certain port drivers from being loaded or to include additional port drivers to load.

Table 6. DVO/SDVO Devices Supported

Device	Legacy VBIOS Support	5.0 VBIOS Support	Graphics Driver Support
Internal LVDS	√	√	√
Chrontel* CH7009/CH7010	√	√	√
Chontel* CH7017		√	√
Chontel* CH7305		√	√
Chrontel* CH7301		√	√
Chrontel* CH7307 (sDVO)		√	√
Chrontel* CH7308 (sDVO)		√	√
Focus* FS453/FS454		√	√
National Semiconductor* NS2501	√	√	√
National Semiconductor* NS387R		√	√
Silicon Image* Sil 164	√	√	√
Silicon Image Sil 1362 (sDVO)		√	√
Silicon Image Sil 1364 (sDVO)		√	√
Thine* Th164	√	√	√

2.2.5 Rotation

Rotation is the ability to rotate the display for the Intel Embedded Graphics Driver. Rotation support includes 0°, 90°, 180°, 270°. Rotation is supported only on the following chipsets using Microsoft Windows NT*, Windows* 2000, Windows XP*, and Linux:

- Intel® 845GV
- Intel® 852GME
- Intel® 852GM
- Intel® 855GME
- Intel® 915GV
- Intel® 915GM
- Intel® 945G
- Intel® 945GM



Rotation is *not* supported in VGA modes of VBIOS, Windows CE*, or the Intel® 815 chipsets, or in conjunction with Direct3D*.

This page intentionally left blank.

Platform Configuration

3

3.1 Configuration Overview

Some aspects of configuring the Intel® Embedded Graphics Drivers are common across the Video BIOS (VBIOS) and the drivers for the supported operating systems. The following sections provide an overview for configuring both the VBIOS and the Intel Embedded Graphics Drivers and describes in detail the common components and tools. This section also describes how to configure the system BIOS for the supported platforms.

3.2 System BIOS Settings

Before installing the Intel Embedded Graphics Drivers, you must first configure the system BIOS for the platform. The following sections describe the required settings. These descriptions are based on AMIBIOS8* from American Megatrends, Inc., which is the recommended system BIOS to use with the Intel Embedded Graphics Drivers. Settings may vary if a different system BIOS is used.

3.2.1 GMCH PCI Device Enabling

The PCI Device Enabling feature on the Graphics and Memory Controller Hub (GMCH) should be set as specified in the following table.

Table 7. GMCH Device 2, Function 1 BIOS Setting

OS	Chipset	
	Intel® 815/815E, Intel® 845GV	Intel® 852x, Intel® 855GME, Intel® 915GV, Intel® 915GM, Intel® 945G, Intel® 945GM
Microsoft Windows* 2000	N/A	Enabled†
Microsoft Windows XP* and Microsoft Windows XPe*	N/A	Disabled
Microsoft Windows CE*	N/A	Disabled
Linux	N/A	Disabled
† Graphics driver is required to be installed for Device 2, Function 0 and Device 2, Function 1.		

3.2.2 Graphics Mode Select (GMS)

The System BIOS typically allows a portion of physical memory to be dedicated to firmware and graphics driver use. This dedicated memory is known as stolen memory since it is not available to the operating system. The size of this memory is selectable and chipset-specific; for example, for the Intel® 845GV chipset, stolen memory can be configured to 0, 512 Kbytes, 1 Mbyte, or 8 Mbytes. Stolen memory is typically used by the firmware and graphics driver to locate the frame

buffer, but can also be used as scratch and surface memory. Since it is programmatically set aside during boot by system BIOS, access to it is direct and does not require OS memory allocation services. Firmware is fully responsible for stolen memory management.

Graphics Mode Select (GMS), or stolen memory, can be set to any of the sizes listed in [Table 8](#). Smaller sizes limit the frame buffer size during firmware boot. Larger sizes marginally increase surface allocation performance for the graphics driver.

Table 8. GMS Settings

Chipset	GMS Settings
815/815E	0, 512 Kbytes, 1 Mbytes
845GV	0, 512 Kbytes, 1 Mbytes, 8 Mbytes
852GM	0, 1 Mbytes, 4 Mbytes, 8 Mbytes, 16 Mbytes, 32 Mbytes
852GME/855GME	0, 1 Mbytes, 4 Mbytes, 8 Mbytes, 16 Mbytes, 32 Mbytes
915GV/915GM	0, 1 Mbytes, 8 Mbytes
945G/945GM	0, 1 Mbytes, 8 Mbytes

3.2.3 AGP (Accelerated Graphics Port) Aperture Size

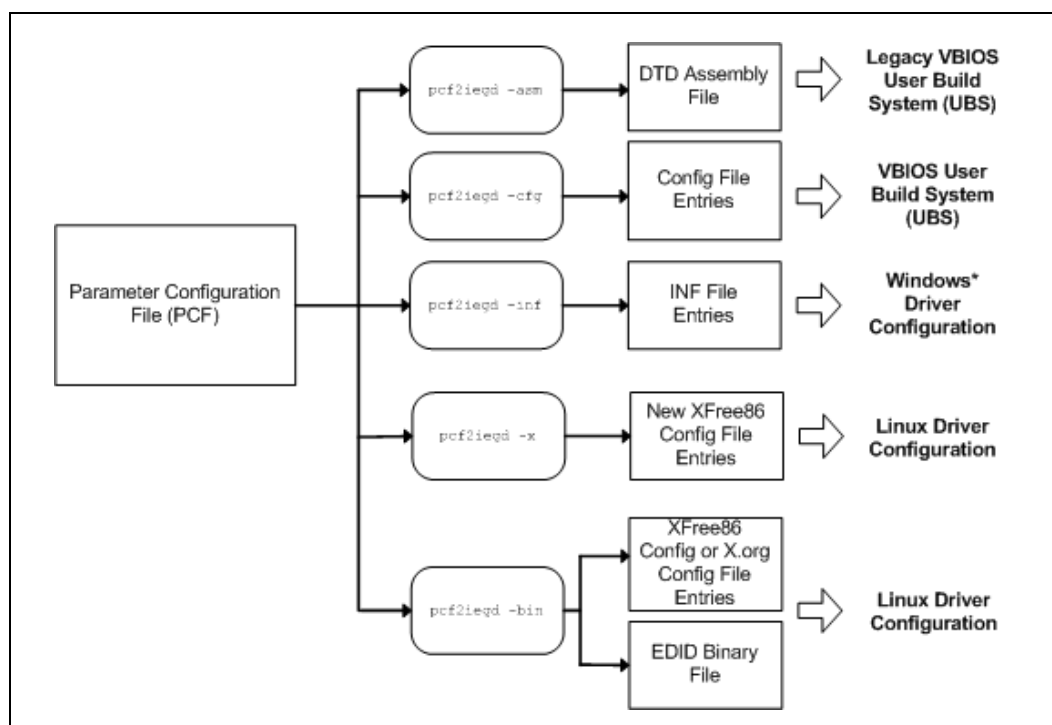
The AGP Aperture size can be set to 64 Mbytes or 128 Mbytes. This controls the total amount of graphics memory that can be mapped in the AGP Aperture.

3.3 VBIOS and Driver Configuration

The Intel Embedded Graphics Suite allows customer configuration of both the VBIOS and graphics driver as well as programming of Detailed Timing Descriptors (DTDs) for EDID-less panels for both the VBIOS and graphics driver. This is accomplished using a single user-readable parameter configuration file (PCF) that contains a table of possible DTDs, each associated with a potential target panel and display mode for the system. This file is processed through a utility called `pcf2iegd` that generates DTD and configuration settings used by the IEGD.

In the case of the VBIOS, the output is in the form of files used by the User Build System (UBS) to compile and link the final firmware image. In the case of the graphics driver, the provided tools convert the PCF format to INF or binary entries suitable for merging into the installation files for the Microsoft Windows* or Linux operating systems. Upon driver installation, the system uses the installation file entries to generate a DTD table in the system registry. [Figure 4](#) illustrates how the `pcf2iegd` utility is used to generate the necessary configuration and DTD files from a single PCF file.

Figure 4. Conversion of PCF for VBIOS and Driver Configuration



3.4 pcf2iegd Usage

The following sections describe the syntax and usage of the `pcf2iegd` utility program.

3.4.1 pcf2iegd -ver

To display the current version of the IEGD, execute the utility with the `-ver` option:

```
pcf2iegd -ver
```

3.4.2 pcf2iegd -h

To display usage and help text, execute the utility with the `-h` option:

```
pcf2iegd -h
```

3.4.3 pcf2iegd -asm

Use the following options to generate a DTD assembly file to be used by the User Build System in configuring the VBIOS. The `<config_id>` is optional, and specifies which of the multiple configurations in the PCF file to use. If not specified, the `pcf2iegd` tool will select the first

configuration in the PCF file. The <input file name> is the name of the PCF file. The output of the utility must be redirected to an output assembly file, <output file name>, so that the contents may be used by the User Build Systems.

```
pcf2iegd -asm [<config_id>] <input file name> > <output file name>
```

Note: The go.bat program calls the pcf2iegd program with the -asm option, so you typically will not run the pcf2iegd program with this option.

3.4.4 pcf2iegd -cfg

Use the -cfg option to generate a configuration file to be used by the User Build System in configuring the 5.0 version of the VBIOS. The <config_id> is optional, and specifies which of the multiple configurations in the PCF file to use as the default. If not specified, the pcf2iegd tool selects the first configuration.

```
pcf2iegd -cfg <input file name> [<config_id>]
```

Note: The go.bat program calls the pcf2iegd program with the -cfg option, so you typically will not run the pcf2iegd program with this option.

3.4.5 pcf2iegd -inf

Use the following options to generate INF entries for configuration of the Microsoft Windows* version of the driver. The <input file name> is the name of the PCF file. The output of the utility must be redirected to an output file, <output file name>. The contents of this file must be inserted into the .inf file that is used when installing the Microsoft Windows* driver. These settings must be inserted in the correct block of the INF file as follows:

For the i81x chipsets, insert these settings in the
[ssigd_SoftwareDeviceSettings_wht] block of the INF file.

For the i845, and i85x, insert these settings in the
[ssigd_SoftwareDeviceSettings_alm] block of the INF file.

For the 915x and 945x, insert these settings in the
[ssigd_SoftwareDeviceSettings_nap] block of the INF file.

```
pcf2iegd -inf <input file name> > <output file name>
```

Note: The -inf option does not currently take a config_id, as do the -bin and -asm options. This is because all configurations are included into a single INF file, per the Universal INF design in the Microsoft Windows* version of the driver. To support multiple configurations and DTDs in the Linux driver, you must execute the pcf2iegd utility for each configuration in the PCF file.

3.4.6 pcf2iegd -x

The -x option generates configurations for the Linux version of the driver. This option is the preferred method for generating Linux configuration information for the XF86conf and Xorg.conf files and replaces the -bin option for Linux configurations. The -x option produces global and per-configuration information that can be cut and pasted into the XF86conf or Xorg.conf file. Note that the -bin option of the pcf2iegd command is still supported.

The <config_id> is optional, and specifies which of the multiple configurations in the PCF file to use. If not specified, the `pcf2iegd` tool selects the first configuration in the PCF file. The <input file name> is the name of the PCF file. The output of the utility must be redirected to an output file, <output file name>. This output is an example of how to configure the blocks of the XF86Conf or Xorg_conf file for use of IEGD in Linux. See [Section 8, “Linux Installation and Configuration” on page 125](#) for more details on the configuration steps for Linux.

```
pcf2iegd -x [<config_id>] <input file name> > <output file name>
```

3.4.7 pcf2iegd -bin

The `-bin` option generates the required configuration files for the Linux version of the driver. The <config_id> is optional, and specifies which of the multiple configurations in the PCF file to use. If not specified, the `pcf2iegd` tool will select the first configuration in the PCF file. The <input file name> is the name of the PCF file. The output of the utility must be redirected to an output file, <output file name>. This output is an example of how to configure the blocks of the XF86Conf or Xorg_conf file for use of IEGD in Linux. The `-bin` option also generates a binary EDID file that is used for the configuration of the Linux driver. See [Section 8, “Linux Installation and Configuration” on page 125](#) for more details on the configuration steps for Linux.

```
pcf2iegd -bin [<config_id>] <input file name> > <output file name>
```

3.5 Parameter Configuration Format

Only one PCF file is needed for VBIOS configuration, Microsoft Windows* graphics driver INF configuration, and Linux graphics driver EDID-less BIN file configuration. This section describes the format and options for the PCF file.

Note: The name of the PCF file you create must not exceed the DOS command line limit of 32 characters including the .pcf suffix. If it exceeds 32 characters, the `pcf2iegd` utility will not process the file and it displays the following error message:

"Pcf2iegd: Error, no input file specified."

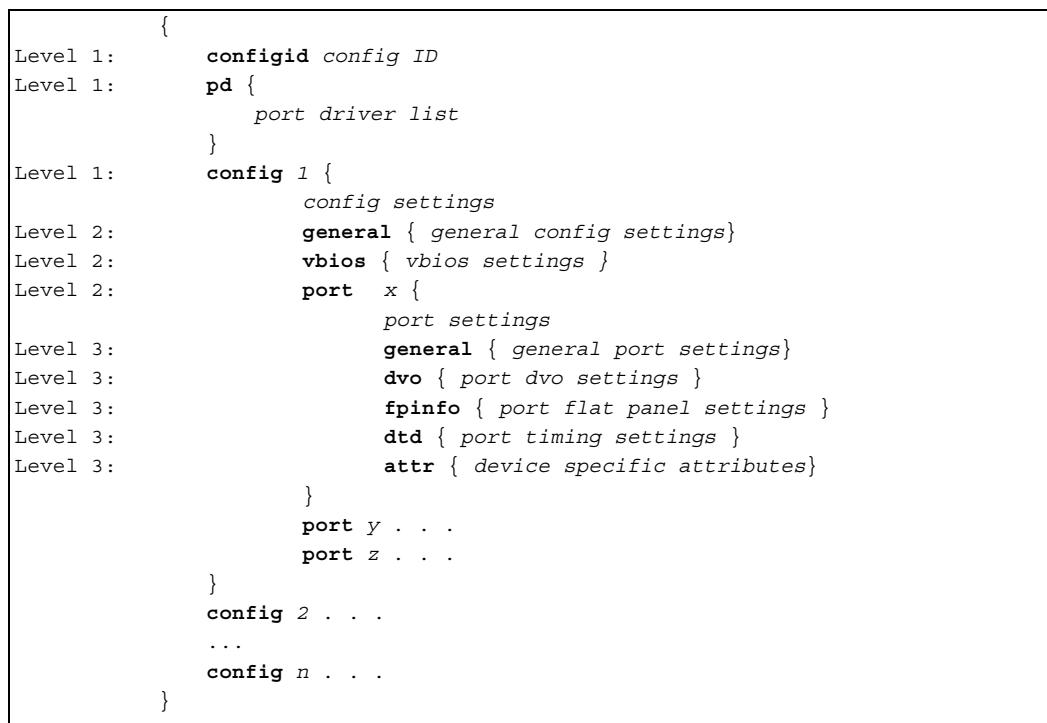
General format rules

1. Comments: Comments can be inserted anywhere in a PCF file.
 - All characters following the '#' will then be ignored by the PCF parser.
2. Blocks: Syntax: [[block_name] [block_id]] { }.
 - Blocks are denoted by open and close braces { }.
 - Blocks may have an optional block_name.
 - Block names may contain an optional block_id.
 - Blocks may contain sub blocks.
 - Blocks may contain assignments, comments, or blank lines.
 - block_name: Non-quoted ascii string
 - block_id: Integer
3. Assignments: Syntax: <name> = <value>

- Name: Non-quoted ascii string.
- Value: Integer or Quoted ascii string.

Table 9 describes the available settings. The gray rows in Table 9 are block headings, and the non-gray rows that follow each heading are settings within that block. Some of these block headings are contained within prior block headings. Figure 5 shows the hierarchy of the block headings.

Figure 5. Hierarchy of PCF File Block Headings



Note: There may be multiple config blocks within a PCF file and multiple port blocks within each config block. The general, vbios, dvo, fpinfo, and dtd blocks should only occur once in each of their corresponding config or port blocks. Also, all config blocks must be within braces at the beginning and end of the PCF file.

Table 9. Parameter Configuration Format (Sheet 1 of 8)

Name	Range/Value	Description
pd	<p>List of port drivers used for the VBIOS. For example: sii164 = 1 ch7009 = 0 ns2501 = 0 etc...</p> <p>(Note that this section does not configure the port driver settings for the IEGD, only the VBIOS)</p>	<p>This parameter enables port drivers for the VBIOS. You can enable as many port drivers as you want but there is a space limitation of 64K available for port drivers. If you exceed this amount of space, the VBIOS will fail to build. You should only enable the port drivers that are required.</p> <p>1 to enable a driver. 0 to disable a driver</p> <p>Note: You can select only one from the following list of port drivers: CH7009, CH7017, FS454</p>
configid	Integer (1-15)	<p>Optional keyword used to specify which configuration the <code>pcf2iegd</code> utility uses. The config ID specified here must match one of the configuration IDs defined within the .pcf file. If this keyword is omitted and you use the <code>-inf</code> option of the <code>pcf2iegd</code> utility, then all configurations specified in the .pcf file are used.</p> <p>Note that this keyword is not required for Linux and vBIOS configurations since the config ID can be specified as a command line argument to the <code>pcf2iedg</code> utility. Refer to Section 3.4, "pcf2iegd Usage" on page 29 for information on using the <code>pcf2iegd</code> utility.</p>
config	Integer (1-15)	More than one configuration is valid.
comment		<p>A quoted string used to identify the origin of the .bin or .inf file.</p> <p>Comment is a required field for Linux configurations.</p>
name		<p>A quoted string used to identify the configuration name.</p> <p>Name is a required field for VBIOS configuration.</p> <p>NOTE: Position of this parameter in the PCF file is critical and will cause errors in the VBIOS settings if moved.</p>
general		Settings that are generic to the configuration.

Table 9. Parameter Configuration Format (Sheet 2 of 8)

Name	Range/Value	Description
displayconfig	1 – Single 2 – Clone 4 – Twin 8 – Extended Default: 8	Used to configure initial state of attached displays. 1 – Single. A single display. 2 – Clone. Primary and secondary displays enabled, configured with separate timing pipes. This allows different timings to be applied to each display. Resolutions must be the same on both displays. 4 – Twin. Primary and secondary displays are enabled, but with only a single pipe. Both displays share the same resolutions and timings. 8 – Extended. Configures separate pipes to allow primary and secondary displays to have different resolutions and display different content. Upon first boot after the driver installation, this option will only enable the primary display, as the extended modes must be enabled in the operating system (i.e., Extended Desktop in the Display Properties sheet within Microsoft Windows).
displaydetect	0 - Disable 1 - Enable	Enable or disable Display Detection. Note that this parameter must be Enabled in order to use COMMON_TO_PORT values. Default is 0. Refer to Section 3.7, "Display Detection and Initialization" on page 47 for detailed information on this parameter.
PortOrder	PortOrder must be specified as a quoted string containing four digits. The valid values are: 1 - Integrated TV Encoder (915GM only) 2 - DVO/sDVO B port 3 - DVO/sDVO C port (or DVO port on 815/E) 4 - Integrated LVDS port (852GME, 855GME, or 915GM only) 5 - Analog CRT port Default: 0 for all keys	Search order for detecting attached displays for the Display Detection feature. When Display Detection is enabled, the PortOrder determines which display is primary and which display is secondary. The port search order can be specified to ensure the port device (DVO device) is found, based on the system integrator's routing choices. Default ordering is chosen by specifying zeros in the PortOrder keys. Default ordering is chipset specific; see Table 32, "Default DVO Search Order" on page 180 . Refer to Section 3.7, "Display Detection and Initialization" on page 47 for more information on using PortOrder in combination with the Display Detect feature.
clonewidth cloneheight	Typical sizes: clonewidth – 800, cloneheight - 600 clonewidth – 1024, cloneheight - 768 clonewidth – 1280, cloneheight - 768 clonewidth – 1400, cloneheight – 1050	Width and height for a cloned display.
clonerefresh = 60	Typical refresh rates (expressed in Hz): 60Hz, 75Hz, 85Hz	Refresh rate for a cloned display.

Table 9. Parameter Configuration Format (Sheet 3 of 8)

Name	Range/Value	Description
vbios		This block contains settings for the new Video BIOS. Note that you only need to specify the parameters you are actually using. You do not need to specify all the parameters in this block. If you omit any parameters, the vbios uses the default values.
COMMON_TO_PORT	6 digit value	Maps the ports from the system BIOS to a port number used by the graphics hardware. Refer to Section 4.3.4, "Configuring the PCF File for the Video BIOS" on page 61 for more information on this parameter. Note that the <code>displaydetect</code> parameter must be set to Enabled in order for the <code>COMMON_TO_PORT</code> values to be used. The default is all zeroes: 000000
post_display_msg	0 - disable greater than 0 - enable and display POST message for the specified number of seconds	Enables or disables the POST (Power On Self Test) message. When you specify a value greater than 0, the message is displayed for the specified number of seconds. For example: <code>post_display_msg = 5</code> This enables the POST message and displays it for approximately 5 seconds. The maximum value that can be entered here is 65535. The default is 1, enable and display the POST message for approximately 1 second.
oem_string	double-quoted string	This string appears on the display when the <code>post_display_msg</code> is enabled and the VBIOS starts up. The maximum string length is 100 characters. The default is " " (two double quotes with a single space in between).
oem_vendor	double-quoted string	This string appears on the display when the <code>post_display_msg</code> is enabled and the VBIOS starts up. The maximum string length is 80 characters. The default is " " (two double quotes with a single space in between).
oem_product_name	double-quoted string	This string appears on the display when the <code>post_display_msg</code> is enabled and the VBIOS starts up. The maximum string length is 80 characters. The default is " " (two double quotes with a single space in between).
oem_product_rev	double-quoted string	This string appears on the display when the <code>post_display_msg</code> is enabled and the VBIOS starts up. The maximum string length is 80 characters. The default is " " (two double quotes with a single space in between).

Table 9. Parameter Configuration Format (Sheet 4 of 8)

Name	Range/Value	Description
int15	5 digits	<p>This parameter allows you to enable or disable the five System BIOS 15h interrupt hooks. The value must be 5 digits in length. Each digit is associated with one of the five System BIOS interrupt 15h hooks as shown below (left to right)</p> <p>1 - 5F31h, POST Completion Notification Hook 2 - 5F33h, Hook After Mode Set 3 - 5F35h, Boot Display Device Hook 4 - 5F36h, Boot TV Format Hook 5 - 5F38h, Hook Before Set Mode</p> <p>(Refer to Appendix D, "Intel 5F Extended Interface Functions" for more information on 5F functions.)</p> <p>The value of each digit must be a 0 or a 1 as follows:</p> <p>0 - disable a System BIOS 15h hook 1 - enable a System BIOS 15h hook</p> <p>For example,</p> <p>int15 = 11001</p> <p>Enables 5F31h, 5F33h, and 5F38h hooks only. The 5F35h and 5F36h hooks are disabled.</p> <p>The default is 11111, enable all five hooks.</p>
default_refresh_0 default_refresh_1	Valid refresh rate in Hz. (specified in decimal, e.g., default_refresh_0=60).	<p>These settings allow you to specify which refresh rate is used for certain VESA modes on the primary and secondary displays. For example, mode 0x117 specifies refresh rates of 60Hz, 75Hz, and 85Hz. This setting allows you to specify which of those three rates to use.</p> <p>The default is 60 for both parameters.</p>
default_mode_0 default_mode_1	Valid VGA or VESA mode. (e.g., default_mode_0 = 3)	<p>These settings establish the default VGA or VESA mode to use for the primary (0) and secondary (1) displays. The values should be set to a valid standard VGA or VESA mode (in hexadecimal format, for example, 0x117). Note that if you select a VGA mode, the secondary display will automatically be set to this mode since the VBIOS can only support one simultaneous VGA mode.</p> <p>The default is mode 3 for both.</p>
default_vga_height	Valid VGA height	<p>This setting allows you to specify which resolution is used for certain VGA modes. Since only one VGA mode can be supported on both displays, this setting applies to the primary display mode (default_mode_0) only. For example, mode 3 specifies three possible resolutions: 640x200, 640x350, and 720x400. In this example, setting default_vga_height=350 indicates the resolution 640x350.</p>

Table 9. Parameter Configuration Format (Sheet 5 of 8)

Name	Range/Value	Description
port	1 - Integrated TV Encoder (915GM only) 2 - DVO/sDVO B port 3 - DVO/sDVO C port (or DVO port on 815/E) 4 - Integrated LVDS port (852GME, 855GME, or 915GM only) 5 - Analog CRT port	Used to define port specific settings.
name		A quoted string used to identify the port.
general		Settings that are generic to the port.
rotation	Windows* Range: 0x0 or 0 – 0 degrees 0x5A or 90 – 90 degrees 0xB4 or 180 – 180 degrees 0x10E or 270 – 270 degrees Linux Range: 0 – 0 degrees 90 – 90 degrees 180 – 180 degrees 270 – 270 degrees Default: 0	Rotation of the display.
flip	Windows: 0x0 or 0 – turn off horizontal flip 0x1 or 1 – turn on horizontal flip Default: 0 Linux Boolean: on - horizontal flip off - no horizontal flip Default: off	Flip of the display.
edid	0 – Do not read EDID from panel/CRT 1 – Attempt to extract EDID timing data from panel/CRT	If VBIOS/Driver reads EDID from panel/CRT.

Table 9. Parameter Configuration Format (Sheet 6 of 8)

Name	Range/Value	Description
edid_avail edid_not_avail	<p>Range [16 bits]</p> <p>Valid values (specified in hex):</p> <p>bit 0 ----- 0 - Do not use driver built-in standard timings 1 - Use driver built-in standard timings</p> <p>bit 1 (not applicable to edid_not_avail) ----- 0 - Do not use EDID block 1 - Use EDID block and filter modes</p> <p>bit 2 ----- 0 - Do not use user-defined DTDs 1 - Use user-defined DTDs</p> <p>bit3 - bit15 ----- Reserved for future use.</p>	<p>These two parameters are used to control the available timings for any display. <code>edid_avail</code> is used when EDID values are read from the display. If an attempt to read EDID from the display fails or the <code>edid</code> parameter is set to 0, then the driver uses the <code>edid_not_avail</code> flags.</p> <p>The value for both parameters must be specified as a hex value.</p> <p>Defaults: <code>edid_avail</code>: 3 (hex). Bit 0 = 1, Bit 1 = 1, Bit 2 = 0 (Use driver built-in standard timings and EDID block and filter modes.) <code>edid_not_avail</code>: 1 (hex). Bit 0 = 1, Bit 1 = 0, Bit 2 = 0. (Use driver-built-in standard timings.)</p> <p>Refer to Section 3.8, "Advanced EDID Configuration" on page 50 for detailed information.</p>
multidvo	<p>0 – Do not attempt to detect a second decoder of same type</p> <p>1 – After detect of a decoder, continue to attempt detection of same type of decoder until fail</p>	<p>If VBIOS/Driver detects a second decoder of same type.</p> <p>This value is hard-coded to "1" for Windows* configuration and will ignore this setting.</p>
dvo		DVO device information.
i2cpin	<0-6>	<p>The GPIO pin pair used on the I2C bus to read and write to DVO device registers.</p> <p>For pin pair definitions, see Table 10, "Pin Pair Definitions" on page 41.</p>
ddcpin	<0-6>	<p>The GPIO pin pair used as DDC bus to read panel EDID data.</p> <p>For pin pair definition, see Table 10, "Pin Pair Definitions" on page 41.</p>
i2cdab	<0x00-0xff>	<p>I2C device address for reading and writing device registers.</p> <p>The device address should be in 8-bit format with the 7-bit slave address assigned to its bits 7:1 and bit 0 set to 0.</p>
ddcdab	<0x00-0xff>	I2C device address for reading EDID data from display through the DDC bus.
i2cspeed	[10-400]. Units in KHz	Speed of I2C bus for DVO device.
ddcspeed	[10-400]. Units in KHz	Speed of I2C bus for EDID device.
fpinfo		Panel-specific information. Some DVO devices require specific programming of a flat panel's width and height.

Table 9. Parameter Configuration Format (Sheet 7 of 8)

Name	Range/Value	Description
width height	Typical values for Width and Height: Width – 800, Height - 600 Width – 1024, Height - 768 Width – 1280, Height - 768 Width – 1400, Height – 1050 Default: 0 for all keys	Native Flat Panel Width and Height.
bklmethod	Range [0-3] 0 – no backlight 1 – Port Driver 2 – GMCH 3 – ICH	Instructs which backlight method is required for the panel attached to the given port. If zero is supplied, or the key is not present, then no backlight control is provided.
bkltt1	Range [0-0xffff]. Units in ms	Time delay between VDD active, and DVO clock/data active. Zero indicates no delay required.
bkltt2		Time delay between DVO clock/data active and Backlight enable.
bkltt3		Time delay between Backlight disable and DVO clock/data inactive.
bkltt4		Time delay between DVO clock/data inactive and VDD inactive.
bkltt5		Minimum delay between VDD inactive, and active.
gpiopinvee	Valid ICH GPIO pin, 0 indexed	GPIO connection for panel power.
gpiopinvee	For example: gpiopinvee = 3 gpiopinvee = 5 gpiopinenable = 1	GPIO connection for backlight power on/off sequencing signal.
gpiopinvee		GPIO to enable backlight signal.
gpiopinvee		
UseGMCHClockPin	1 - Flat panel is connected to the clock pin 0 - Flat panel is not connected to the clock pin	This entry is needed when GMCH is selected as backlight control method.
UseGMCHDataPin	1 - Flat panel is connected to the data pin 0 - Flat panel is not connected to the data pin	This entry is needed when GMCH is selected as backlight control method.
dtd		Denotes a Detailed Timing Descriptor (DTD) block. Settings in this section, except for the <code>flags</code> parameter, correspond to the Detailed Timing Block described in the VESA standard "Extended Display Identification Data Standard", Version 3, November 13, 1997.
p_clock	Range [0-0x7ffffff]	Pixel clock value in KHz.
h_active	Range 0-4096 [12 bits]	Horizontal Active.
v_active	Range 0-4096 [12 bits]	Vertical Active.
h_sync	Range 0-1024 [10 bits]	Horizontal Sync Offset.
v_sync	Range 0-64 [6 bits]	Vertical Sync Offset.
h_syncp	Range 0-1024 [10 bits]	Horizontal Sync Pulse Offset.
v_syncp	Range 0-64 [6 bits]	Vertical Sync Pulse Width.

Table 9. Parameter Configuration Format (Sheet 8 of 8)

Name	Range/Value	Description
h_blank	Range 0-4096 [12 bits]	Horizontal Blanking.
v_blank	Range 0-4096 [12 bits]	Vertical Blanking.
h_border	Range 0-256 [8 bits]	Horizontal Border. Currently not supported.
v_border	Range 0-256 [8 bits]	Vertical Border. Currently not supported.
h_size	Range 0-4096 [12 bits]	Horizontal Size. Currently not supported.
v_size	Range 0-4096 [12 bits]	Vertical size. Currently not supported.
flags	Range [32 bits] Valid values: bit 31 ----- 0 - Non-interlaced 1 - Interlaced bit 27 ----- 0 - vertical sync polarity active low 1 - vertical sync polarity active high bit 26 ----- 0 - horizontal sync polarity active low 1 - horizontal sync polarity active high bit 25 ----- 0 - blank sync polarity active high 1 - blank sync polarity active low bit 17 ----- 0 - Normal DTD 1 - Panel/display Native DTD All other bits ----- Do not use any other bits; all other bits must be set to 0.	Interlace, Horizontal polarity, Vertical polarity, Sync Configuration, etc. Note that these flags are IEGD specific and do not correspond to VESA 3.0 flags. For example, to set Interlaced with Horizontal Sync Polarity high (bits 31 and 26), then the flags value = 0x84000000. (Binary = 10000100000000000000000000000000)
attr	0-0xFFFF	Attribute values that are specific to the DVO device for the port. See Appendix C, "Port Driver Attributes" for specific attribute IDs and associated values.
id <Attribute ID>	0 - 2 ³²	id = <value>. Both the Attribute ID and its value should be specified in decimal. For example, to set brightness to 50, you specify id 0 = 50

Table 10. Pin Pair Definitions

Pair #	Signal Name	Signal Description	Notes	815/E Pin Names	845GV Pin Names	852GME 855GME Pin Names	915GV Pin Name	915GM Pin Names
0	DDCA	CRT DDC for Analog monitor (CRT) connection.	This cannot be shared with other DDC or I2C pairs due to legacy monitor issues.	DDCK DDDA	DDCA_CLK DDCA_DATA	DDCACLK DDCADATA	DDCCLK DDCDATA	DDCCLK DDCDATA
1	LCLKCTL	SCC Chip - For control of SSC clock generator devices down on motherboard.	If SSC is not supported then can be used for DVOB or DVOC GMBUS.	LTVCK LTVDA		LCLKCTLA LCLKCTRLB		LCTLA_CLK LCTLB_DATA
2	DDCP	Panel DDC for Digital Display connection via the integrated LVDS display port for support for EDID panel.	If EDID panels are not supported. Can optionally use as GMBUS for DVOB or DVOC.			DDCPCLK DDCPDATA		LDDC_CLK LDDC_DATA
3	MDVI	DVI 1 DDC - GMBUS control of DVI devices (TMDS or TV encoder)	Can optionally use as GMBUS for DVOB or DVOC.		MDVI_CLK MDVI_DATA	MDVICLK MDVIDATA		
4	MI2C	DVO I2C - GMBUS control of DVI devices (TMDS or TV encoder)	Can optionally use as GMBUS for DVOB or DVOC.		MI2C_CLK MI2C_DATA	MI2CCLK MI2CDATA	SDVOCTRL_CLK SDVOCTRL_DATA	SDVOCTRL_CLK SDVOCTRL_DATA
5	MDDC	DVI 2 DDC - DDC for Digital Display connection via TMDS device	Can optionally use as GMBUS for DVOB or DVOC.		MDDC_CLK MDDC_DATA	MDDCCLK MDDCDATA		
6	EXTTS	External Thermal Sensor Input				EXTTS_0		

3.6 Sample PCF File

Below is a sample PCF file.

```
#Sample PCF File
#-----
# Do not Edit lines beginning with [
{
    # -----
    # Maximum of 2 port drivers may be selected
    # 1 to enable, 0 to disable
    # -----
    pd {
        sii164 = 1
        ch7017 = 1
    }

    config 1 {
        general {
            portorder = "3500"

            # Setup the 64MB memory-usage display
            displayconfig = 4
            # 1 to enable, 0 to disable
            displaydetect = 1
        }
        vbios {
            # This is a mapping from common name (and names used in some legacy 5f
            # functions) to IGD port numbers
            # It should be configured based on the hardware platform.
            #
            # 5f Names (Per Spec)
            # 0 : CRT
            # 1 : TV1
            # 2 : Flat Panel (DVI)
            # 3 : Local Flat Panel
            # 4 : TV2
            # 5 : Flat Panel 2 (DVI)
            #
            # The array index is the 5f bit number. The value is the IGD port number.
            COMMON_TO_PORT=503000

            # 1 to enable, 0 to disable
            post_display_msg=1

            oem_string="Intel(r) Graphics Chip Embedded VGA BIOS"
            oem_vendor_name=" "
            oem_product_name="IEGD"
            oem_product_rev="5.0"

            #vBIOS supports max of 2 displays. That means either 2 displays on one
            #pipe or two pipes each with one display.

            default_refresh_0=47
            default_refresh_1=47

            default_mode_0=3
            default_mode_1=3

            default_vga_height=400
        }
    }
    port 3 {
        name = "DVO-C"
        general {
            rotation = 0
            # 1 to enable, 0 to disable
            edid = 1
            # 1 to enable, 0 to disable
            edid_avail = 0x3    # Flags when EDID is available
            edid_not_avail = 0x1 # Flags when EDID is not available
            multidvo = 0
        }
        dvo {
```

```

        ddcpin = 4
        ddcspeed = 100
        i2cpin = 4
        i2cspeed = 100
    }
    dtd 1 {
        p_clock = 30000
        h_active = 800
        h_sync = 32
        h_syncp = 120
        h_border = 0
        h_blank = 224
        v_active = 600
        v_sync = 3
        v_syncp = 2
        v_border = 0
        v_blank = 24
        h_size = 0
        v_size = 0
        flags = 0x18
    }
    attr {
        id 9 = 1
    }
}
port 5 {
    name = "Analog"
    general {
        rotation = 0
        # 1 to enable, 0 to disable
        edid = 1
        # 1 to enable, 0 to disable
        multidvo = 0
    }
}
}
config 2 {
    name = "VGA"
    general {
        portorder = "3500"
        #
        # Setup the 64MB memory-usage display
        displayconfig = 4
        # 1 to enable, 0 to disable
        displaydetect = 1
    }
    vbios {
        # This is a mapping from common name (and names used in some legacy 5f
        # functions) to IGD port numbers
        # It should be configured based on the hardware platform.
        #
        # 5f Names (Per Spec)
        # 0 : CRT
        # 1 : TV1
        # 2 : Flat Panel (DVI)
        # 3 : Local Flat Panel
        # 4 : TV2
        # 5 : Flat Panel 2 (DVI)
        #
        # The array index is the 5f bit number. The value is the IGD port number.
        COMMON_TO_PORT=503000
        # 1 to enable, 0 to disable
        post_display_msg=1
        oem_string="Intel(r) Graphics Chip Embedded VGA BIOS"
        oem_vendor_name=""
        oem_product_name="IEGD"
        oem_product_rev="5.0"
        #vBIOS supports max of 2 displays. That means either 2 displays on one
        #pipe or two pipes each with one display.
        default_refresh_0=60
        default_refresh_1=60
    }
}

```

```

        default_mode_0=3
        default_mode_1=3

        default_vga_height=350
    }
    port 3 {
        name = "DVO-C"
        general {
            rotation = 0
            # 1 to enable, 0 to disable
            edid = 1
            # 1 to enable, 0 to disable
            edid_avail = 0x3 # Flags when EDID is available
            edid_not_avail = 0x1 # Flags when EDID is not available
            multidvo = 0
        }
        dvo {
            ddcpin = 4
            ddcspeed = 100
            i2cpin = 4
            i2cspeed = 100
        }
        dtd 1 {
            p_clock = 25000
            h_active = 640
            h_sync = 8
            h_syncp = 128
            h_border = 0
            h_blank = 160
            v_active = 480
            v_sync = 10
            v_syncp = 2
            v_border = 0
            v_blank = 45
            h_size = 0
            v_size = 0
            flags = 0x18
        }
        attr {
            id 9 = 1
        }
    }
}
port 5 {
    name = "Analog"
    general {
        rotation = 0
        # 1 to enable, 0 to disable
        edid = 1
        # 1 to enable, 0 to disable
        multidvo = 0
    }
}
}
config 3 {
    general {
        portorder = "3500"

        # Setup the 64MB memory-usage display
        displayconfig = 4
        displaydetect = 1
    }
    vbios {
        # This is a mapping from common name (and names used in some legacy 5f
        # functions) to IGD port numbers
        # It should be configured based on the hardware platform.
        #
        # 5f Names (Per Spec)
        # 0 : CRT
        # 1 : TV1
        # 2 : Flat Panel (DVI)
        # 3 : Local Flat Panel
        # 4 : TV2
        # 5 : Flat Panel 2 (DVI)
    }
}

```

```
#
# The array index is the 5f bit number. The value is the IGD port number.
COMMON_TO_PORT=503000

# 1 to enable, 0 to disable
post_display_msg=1

oem_string="Intel(r) Graphics Chip Embedded VGA BIOS"
oem_vendor_name=" "
oem_product_name="IEGD"
oem_product_rev="5.0"

#vBIOS supports max of 2 displays. That means either 2 displays on one
#pipe or two pipes each with one display.

default_refresh_0=47
default_refresh_1=47

default_mode_0=3
default_mode_1=3

default_vga_height=400
}
port 3 {
    name = "DVO-C"
    general {
        rotation = 0
        # 1 to enable, 0 to disable
        edid = 1
        # 1 to enable, 0 to disable
        edid_avail = 0x3 # Flags when EDID is available
        edid_not_avail = 0x1 # Flags when EDID is not available
        multidvo = 0
    }
    dvo {
        ddcpin = 4
        ddcspeed = 100
        i2cpin = 4
        i2cspeed = 100
    }
    dtd 1 {
        p_clock = 30000
        h_active = 800
        h_sync = 32
        h_syncp = 120
        h_border = 0
        h_blank = 224
        v_active = 600
        v_sync = 3
        v_syncp = 2
        v_border = 0
        v_blank = 24
        h_size = 0
        v_size = 0
        flags = 0x18
    }
    attr {
        id 9 = 1
    }
}
port 5 {
    name = "Analog"
    general {
        rotation = 0
        # 1 to enable, 0 to disable
        edid = 1
        # 1 to enable, 0 to disable
        multidvo = 0
    }
}
}
config 4 {
    general {
        portorder = "3500"
    }
}
```

```

# Setup the 64MB memory-usage display
# 1 to enable, 0 to disable
    displaydetect = 1
}
vbios {
    # This is a mapping from common name (and names used in some legacy 5f
    # functions) to IGD port numbers
    # It should be configured based on the hardware platform.
    #
    # 5f Names (Per Spec)
    # 0 : CRT
    # 1 : TV1
    # 2 : Flat Panel (DVI)
    # 3 : Local Flat Panel
    # 4 : TV2
    # 5 : Flat Panel 2 (DVI)
    #
    # The array index is the 5f bit number. The value is the IGD port number.
    COMMON_TO_PORT=503000

# 1 to enable, 0 to disable
    post_display_msg=1

    oem_string="Intel(r) Graphics Chip Embedded VGA BIOS"
    oem_vendor_name=""
    oem_product_name="IEGD"
    oem_product_rev="5.0"

    #vBIOS supports max of 2 displays. That means either 2 displays on one
    #pipe or two pipes each with one display.

    default_refresh_0=60
    default_refresh_1=60

    default_mode_0=3
    default_mode_1=3

    default_vga_height=400
}
port 3 {
    name = "DVO-C"
    general {
        rotation = 0
        # 1 to enable, 0 to disable
        edid = 1
        # 1 to enable, 0 to disable
        edid_avail = 0x3 # Flags when EDID is available
        edid_not_avail = 0x1 # Flags when EDID is not available
        multidvo = 0
    }
    dvo {
        ddcpin = 4
        ddcspeed = 100
        i2cpin = 4
        i2cspeed = 100
    }
    dtd 1 {
        p_clock = 33550
        h_active = 852
        h_sync = 31
        h_syncp = 6
        h_border = 1
        h_blank = 180
        v_active = 480
        v_sync = 18
        v_syncp = 1
        v_border = 0
        v_blank = 585
        h_size = 0
        v_size = 0
        flags = 0x1E
    }
    attr {
        id 9 = 1
    }
}

```

```

    }
    port 5 {
        name = "Analog"
        general {
            rotation = 0
            # 1 to enable, 0 to disable
            edid = 1
            # 1 to enable, 0 to disable
            multidvo = 0
        }
    }
}

```

3.7 Display Detection and Initialization

The Display Detection and Initialization feature, when enabled, automatically detects displays and allocates ports without the need to change any configuration files. This feature is off by default and can be enabled either through the .pcf file or by directly editing the `ssigd.inf` file for Microsoft Windows* or the `XF86.conf/Xorg.conf` file for Linux.

To enable the feature in the .pcf file, set the `DisplayDetect` parameter to 1 in the `general` section of the `config` block. Refer to [Section 3.5, “Parameter Configuration Format”](#) on page 31 for more information.

Alternatively, you can enable the feature in Microsoft Windows* by entering the following line in the `[ssigd_SoftwareDeviceSettings]` section of the `ssigd.inf` file:

HKR, Config\<ConfigID>\General, DisplayDetect, %REG_DWORD%, 1

where `<ConfigID>` is the configuration ID (without the angle brackets).

To enable the feature in Linux, enter the following line Option setting in the `XF86.conf` or `Xorg.conf` file:

Option "DisplayDetect" "1"

When the display detection feature is enabled, ports are allocated only when the display satisfies the following conditions:

1. The port is not in use (that is, it is not already allocated)
2. The display is detected by the port driver.

The first port that passes these conditions is allocated. If condition 2 fails for all ports, then the first port in the `PortOrder` setting that passes condition 1 is allocated. If the port is not detectable (specifically the internal LVDS or external LVDS using CH7017), the driver assumes the display is connected. Condition number 2 always passes for these displays.

When this feature is disabled, display allocation is done based on `PortOrder` and no display detection is performed.

3.7.1 Display Detect Operation

This section describes the logic of the Display Detection feature and provides several examples.

1. If Display Detect is disabled in the .pcf file, the driver uses the first two ports identified in the PortOrder.
2. If Display Detect is enabled and you are using the 5.0 version of the VBIOS, the VBIOS performs the display detection. The driver then checks to see if the VBIOS returns the display allocations and if it does, the driver does not re-execute the display detection steps.

If you are using the Legacy VBIOS, then the driver performs display discovery as described in the following steps.

3. The number of displays to be detected is based on the DisplayConfig settings in the .pcf configuration file. If this is set to Single, then only one display is detected. If it is set to any other value, a maximum of two displays will be detected.
4. The IEGD goes through each port in the PortOrder settings and attempts to detect a display using the following algorithm:
 - a. If a display is detected, it is based on the PortOrder sequence. Display allocation of the port is performed once the display has been detected. For example:

PortOrder = "5324" (CRT, DVOC, DVOB, LVDS)
Displays Connected = DVOB, CRT

Primary display allocation: searches for a display connected according to the PortOrder sequence. The first detected display is a CRT, so the Primary display is CRT.

Secondary display allocation: Searches for a display connected according to the PortOrder sequence. The first detected and non-allocated display is DVOB, so the Secondary display is DVOB.

- b. If no display is detected on any of the ports, then the driver enables the first available port in the PortOrder. For example:

PortOrder = "5320" (CRT, DVOC, DVOB)
Displays Connected = None

Primary display allocation: Searches for the display connected according to the PortOrder sequence. Since no display is detected, the display allocation is done based on PortOrder. Consequently, the Primary display is set to CRT.

Secondary display allocation: Searches for a display connected according to the PortOrder sequence. Since no display is detected, display allocation is performed based on the PortOrder. The Secondary display is set to DVOC.

- c. The driver cannot detect the presence of a display connected to the Internal LVDS and external LVDS displays connected to some DVO devices (for example, an LVDS connected to the CH7017). Consequently, the driver assumes that an LVDS display is connected if it is in the PortOrder. If you only want to use the internal LVDS when no CRT and DVO devices are connected, then put LVDS in the PortOrder after them. For example:

PortOrder = "5324" (CRT, DVOC, DVOB, LVDS)
Display Connected = None

Primary display allocation: Searches for a display connected according to PortOrder sequence. Since no display is connected and since LVDS is specified in the PortOrder, the driver assumes that an LVDS display is connected. Consequently, the Primary display is set to LVDS.

Secondary display allocation: Searches for a connected display according to the PortOrder. Since no display is connected, the display allocation is performed based on the PortOrder. In this case, the Secondary display = CRT.

- d. Since the driver cannot detect the presence of a display connected to the Internal LVDS and certain external LVDS displays, it therefore always assumes that they are connected if they are listed in the PortOrder. Be careful not to set the PortOrder that prevents the driver from detecting a connected display. For example:

PortOrder = "5432" (CRT, LVDS DVOC, DVOB)
Displays Connected = CRT, DVOC

Primary display allocation: Searches for a connected display according to the PortOrder. In this case, the Primary display is set to CRT.

Secondary display allocation: Searches for a connected display according to the PortOrder. Even though DVOC is connected, the driver assumes that the internal LVDS is also connected. Consequently, the driver never detects the display connected to the DVOC port. To change this, move DVOC before LVDS in the PortOrder ("5342" rather than "5432").

3.7.2 Detectable Displays

Table 11 provides a list of displays that are detectable by the IEGD.

Table 11. Detectable Displays

Transmitter	Display Type	Detectable by IEGD?
GMCH Analog CRT	VGA	Yes
GMCH Integrated LVDS	LVDS	No
GMCH Integrated TV Out	TV Out	N/A (Note: TV out not supported in IEGD.)
CH7009	DVI	Yes
CH7009	TV Out	Yes
CH7017	LVDS	No
CH7017	TV Out	Yes
CH7307	DVI	Yes
CH7308	LVDS	No
SiI164	DVI	Yes
SiI1362	DVI	Yes
SiI1364	DVI	Yes
TH164	DVI	Yes
NS2501	LVDS	Yes
NS387R	LVDS	Yes
FS454	TV Out	Yes

3.8 Advanced EDID Configuration

The `edid_avail` and `edid_not_avail` PCF parameters control the available timings for any display. The `edid_avail` parameter is used when EDID information is read from the display. If the driver is unable to read EDID information from the display or if the `edid` parameter in the `.pcf` file is set to "0" (disable), then the settings of the `edid_not_avail` parameter are used.

The default behavior of `edid_avail` is to use the driver's built-in standard timings and EDID block and filter modes. The default for `edid_not_avail` is to use the driver's built-in standard timings. Refer to [Table 9](#) for more information on these parameters.

The IEGD supports three different types of EDID display modes:

1. **Built-in display modes.** These modes are hard-coded in the IEGD. These modes can be filtered based on the EDID block.
2. **EDID-DTDs:** These are Detailed Timing Descriptors read from the EDID block. EDID can have these DTDs along with other information about the display.
3. **User-specified DTDs** defined in the PCF file.

The Advanced EDID Configuration supports different possible combinations of display modes when an EDID display is present along with user-specified DTDs in the `.pcf` file.

3.8.1 Sample Advanced EDID Configurations

[Table 12](#) presents various EDID configurations and the EDID settings in the `.pcf` file used for those configurations.

Table 12. Sample Advanced EDID Configurations

Configurations	PCF Parameter Settings	Description
1. Use only filtered built-in and any EDID-DTDs when the display has EDID information 2. Use all built-in modes when the display doesn't contain EDID information	<code>edid = 1</code> <code>edid_avail = 3</code> <code>edid_not_avail = 1</code>	Default values.
1. Use only filtered built-in modes and EDID-DTDs when the display has EDID. 2. Use only user-DTDs otherwise.	<code>edid = 1</code> <code>edid_avail = 3</code> <code>edid_not_avail = 4</code>	This configuration allows the IEGD to use its built-in display modes and the modes provided by the display. If the IEGD is unable to read EDID information from the display, then the IEGD uses the user-DTDs defined in the <code>.pcf</code> file.

Table 12. Sample Advanced EDID Configurations

Configurations	PCF Parameter Settings	Description
1. Use only user-DTDs regardless of connected display. (Typically used for a custom panel that only supports user-defined DTDs.) 2. Use limited set of timings when a panel EDID is present, but the IEGD cannot read the EDID information.	edid = 0 edid_avail = (any value) edid_not_avail = 4	Only user-DTDs defined in the .pcf file are used.
1. Use EDID-DTDs for an EDID display. 2. Use user-DTDs for a non-EDID display.	edid = 1 edid_avail = 2 edid_not_avail = 4	This configuration uses the EDID-DTDs when an EDID display is detected and EDID information is read from the display. If the driver detects a non-EDID display, then the IEGD uses user-DTDs defined in .pcf file.
1. Use only EDID-DTDs and user-DTDs for an EDID display. 2. Use user-DTDs only for a non-EDID display.	edid = 1 edid_avail = 5 edid_not_avail = 4	This configuration uses both EDID-DTDs and user-DTDs when the IEGD detects an EDID display. If the driver detects a non-EDID display, then the IEGD uses user-DTDs defined in .pcf file.

3.9 External PCI Graphics Adaptor as Primary Device

The IEGD can be configured to work with an external PCI graphics adaptor card as the primary graphics adaptor device with the Intel internal graphics device (GMCH) as the secondary graphics device. You can configure your system to boot with a PCI graphics adaptor in the System BIOS. When an external PCI graphics adaptor is designated as the primary graphics adaptor, the Intel GMCH becomes the secondary graphics device.

Note: The term *secondary* adaptor refers to the adaptor that is not the *boot-up*, or VGA-Compatible, adaptor. The secondary adaptor is not necessarily the secondary display as assigned by the OS.

You can configure an external PCI card to work with the IEGD as follows:

- The external PCI card as the primary graphics adaptor and the GMCH internal graphics device as the secondary.
- The external PCI card as the secondary graphics adaptor and the GMCH internal graphics device as the primary.

Note: This feature is not supported on Microsoft Windows* CE .NET nor Microsoft Windows NT systems.

The IEGD allows you to specify which display is primary, secondary, and tertiary. It allows Twin and Clone configurations on the internal graphics device when the external PCI display is the primary graphics adaptor. It also allows Twin and Clone configurations on the internal graphics device when the external PCI device is the secondary graphics adaptor.

An external PCI graphics driver runs independently without sharing resources with the IEGD.

The following figures show several configurations when an external PCI adaptor is the primary graphics device and when it is the secondary graphics device.

Figure 6 shows an External PCI card as the primary graphics adaptor card and the IEGD driver as the secondary. The drivers do not share hardware resources. The OS decides the framebuffer content and handles that by drawing to the respective driver independently.

Figure 6. External PCI Graphics Card as Primary Driver and IEGD as Secondary Driver

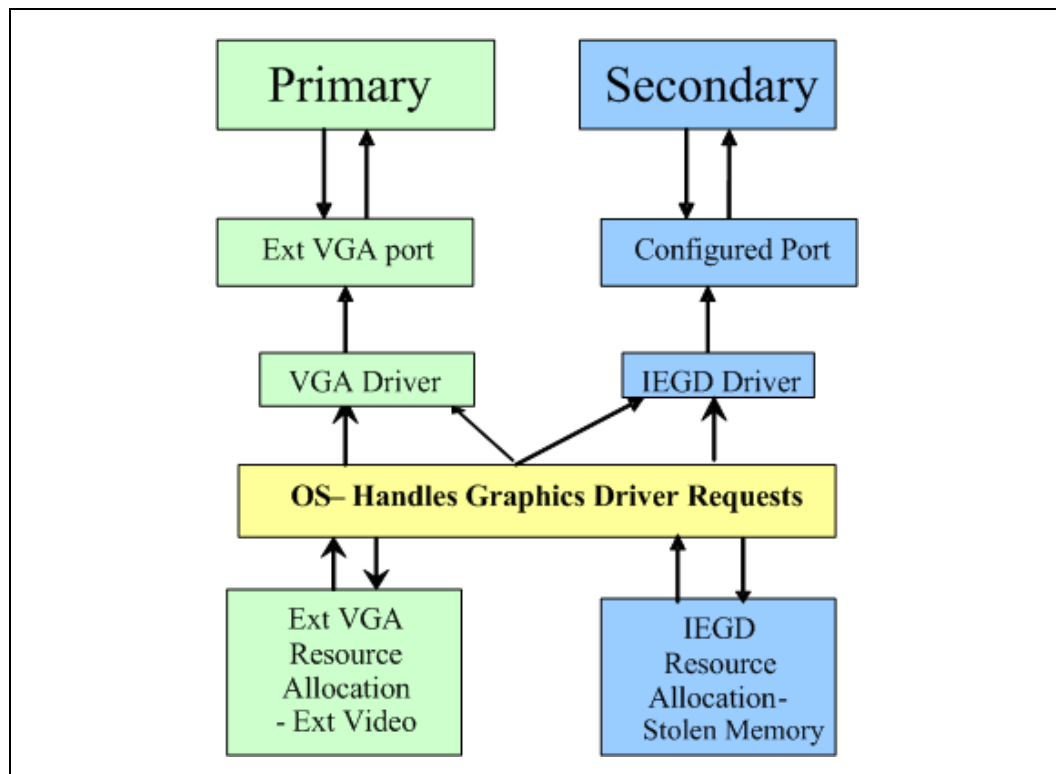


Figure 7 shows the interaction between the IEGD driver and the External VGA driver when the IEGD is booted as the primary driver. Again, the drivers do not share hardware resources. The OS decides the framebuffer content and handles it by drawing to the respective driver independently.

Figure 7. IEGD as Primary Driver and External PCI Graphics Card as Secondary Driver

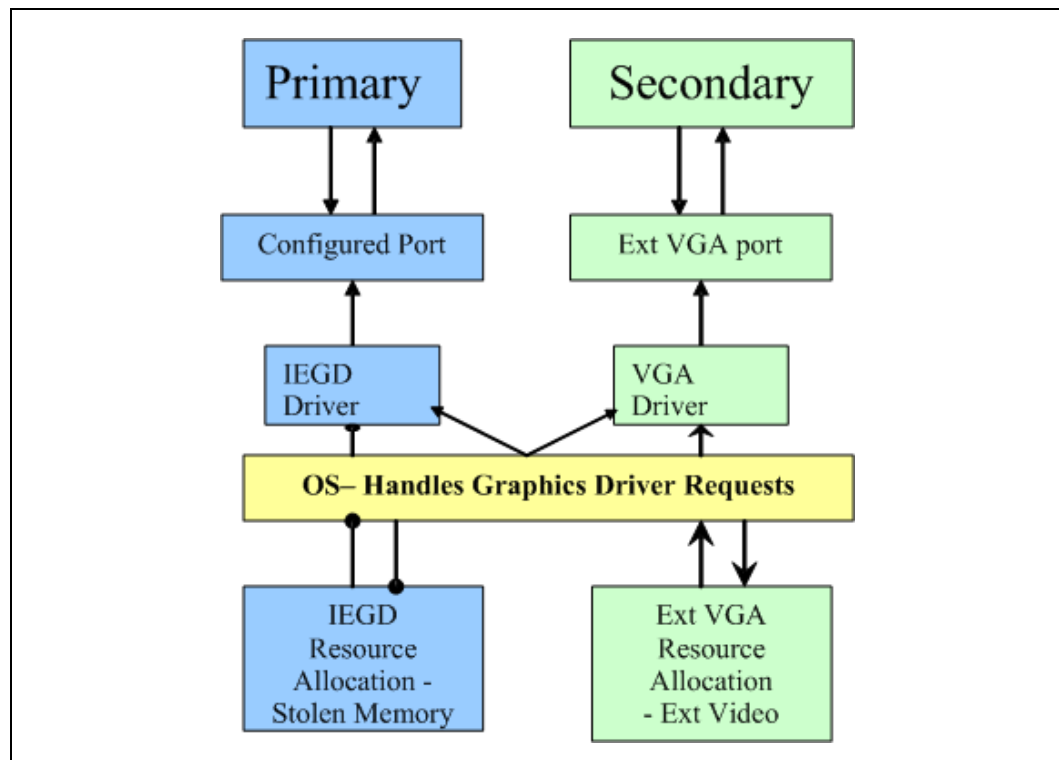
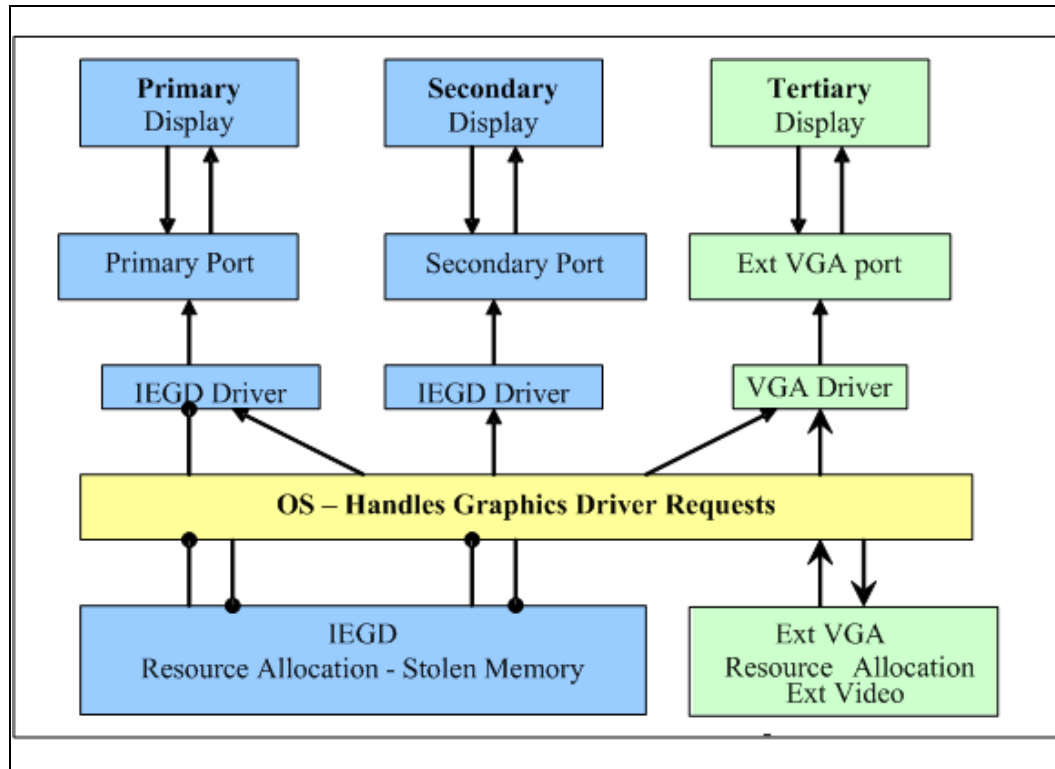


Figure 8 shows a sample configuration where the internal graphics device is primary and configured to use two ports to drive two displays while an external PCI graphics adaptor is used to drive a tertiary display. Note that regardless of the number of ports being assigned to a driver, the external PCI graphics run independently without sharing resources with the IEGD driver.

Figure 8. IEGD as Primary Driver With Two Displays and External PCI Driving a Tertiary Display



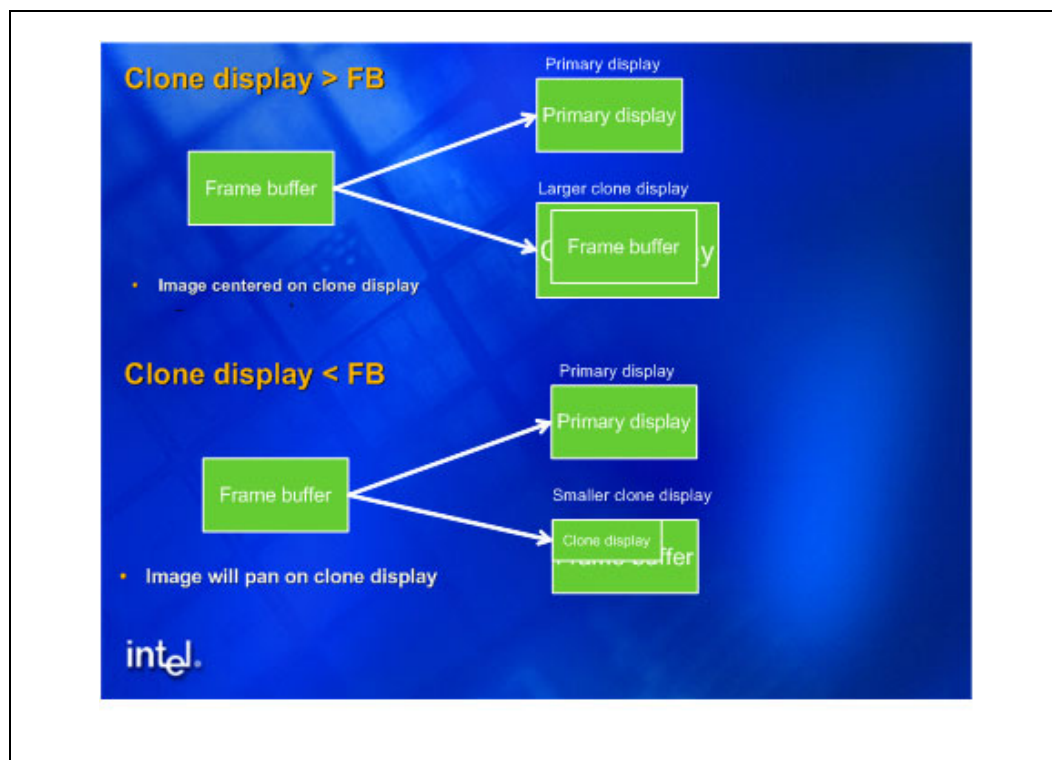
3.10 Enhanced Clone Mode Support

The Enhanced Clone Mode feature allows you to specify a clone display size that is different from the primary display. It also allows you to change the clone display size at runtime using the IEGD Runtime GUI (see [Section 6.6, “Viewing and Changing the Driver Configuration From Microsoft Windows*”](#) on page 95 or [Section 8.5, “Run Time Operation”](#) on page 143 for Linux systems).

In clone mode, the framebuffer is always allocated to match primary display size. On the clone display (secondary display) the image is centered if the display is bigger than the framebuffer. Centering is done only if the requested resolution and refresh rate is not available for the clone display.

[Figure 9](#) shows samples of when the display size of the clone display is different from the framebuffer size.

Figure 9. Extended Clone Mode Displays



Extended clone mode is implemented through the use of three new PCF parameters:

- clonewidth -- allows you to specify a width for the clone display
- cloneheight -- allows you to specify a height for the clone display
- clonerefresh -- allows you to specify a refresh rate for the clone display

3.10.1 Sample Extended Clone Mode PCF File

The following sample PCF file illustrates several example PCF configurations using Extended Clone Mode.

```
# PCF file for enhanced clone mode with
# following combinations:
# 1. CRT + TV
# 2. CRT + integrated LVDS
# 3. CRT + Fixed size DVI display
# 4. CRT + external fixed size LVDS display
# 5. CRT + external fixed size LVDS display connected to scalable encoder.
# 6. CRT + Fixed size DVI display with rotation on
#
#
{
    config 1 {
        #
        # 855 + CH7009 + CRT + TV
        # Choose a CRT that supports bigger than 1024x768 resolution.
        # 1. Set 800x600 on CRT and check 800x600 image appeared
```

```

# at top,left corner on TV.
# 2. Set 1280x1024 on CRT and check TV display is panning.
# Check the clone mouse pointer is in sync with primary display
# mouse pointer.
#
name = "Enhanced clone mode -- 1 CRT+TV"
general {
    displayconfig = 2
    portorder = "5340"
    clonewidth = 1024
    cloneheight = 768
    clonerefresh = 60
    portdrivers = "ch7009"
}

port 3 {
    attr {
        id 9 = 2
    }
}
}

config 2 {
#
# 855 + IntLVDS + CRT + LVDS
# Choose a CRT that supports bigger than 1024x768 resolution.
# 1. Set 800x600 on CRT and check 800x600 image appeared
# at top,left corner on LVDS. Or Image should be scaled to
# match the panel size based on panel used.
# 2. Set 1280x1024 on CRT and check LVDS display is panning.
# Check the clone mouse pointer is in sync with primary display
# mouse pointer.
#
name = "Enhanced clone mode -- 2 CRT+LVDS"
general {
    displayconfig = 2
    portorder = "5430"
    clonewidth = 1024
    cloneheight = 768
    clonerefresh = 60
    portdrivers = "lvds"
}

port 4 {
    fpinfo {
        width = 1024
        height = 768
    }
}
}

config 3 {
#
# 915GV + CH7307 + CRT + DVI
# Choose a CRT that supports bigger than 1024x768 resolution.
# 1. Set 800x600 on CRT and check 800x600 image appeared
# at top,left corner on DVI panel. Or Image should be scaled to
# match the panel size based on DVI panel used.
# 2. Set 1280x1024 on CRT and check DVI display is panning mode.
# Check the clone mouse pointer is in sync with primary display
# mouse pointer.
#
name = "Enhanced clone mode -- 3 CRT+DVI"
general {
    displayconfig = 2
    portorder = "5240"
    clonewidth = 1024
    cloneheight = 768
    clonerefresh = 60
    portdrivers = "sdvo"
}

# set FPINFO so that DVI modes are limited to that size.
port 3 {
    fpinfo {
        width = 1024
        height = 768
    }
}

```



```

    }
}
config 4 {
#
# 855 + NS387 + CRT + LVDS
# Choose a CRT that supports bigger than 1024x768 resolution.
# 1. Set 800x600 on CRT and check 800x600 image appeared
# at top,left corner on LVDS. Or Image should be scaled to
# match the panel size based on LVDS panel used.
# 2. Set 1280x1024 on CRT and check LVDS display is panning mode.
# Check the clone mouse pointer is in sync with primary display
# mouse pointer.
#
name = "Enhanced clone mode -- 4 CRT+Ext.LVDS"
general {
displayconfig = 2
portorder = "5340"
clonewidth = 1024
cloneheight = 768
clonerefresh = 60
portdrivers = "ns387"
}

# set FPINFO so that external LVDS modes are limited to that size.
port 3 {
fpinfo {
width = 1024
height = 768
}
}
}
config 5 {
#
# 855 + NS2501 + CRT + LVDS
# Choose a CRT that supports bigger than 1024x768 resolution.
# 1. Set 800x600 on CRT and check 800x600 image appeared
# at top,left corner on LVDS. Or Image should be scaled to
# match the panel size based on LVDS panel used.
# 2. Set 1280x1024 on CRT and check LVDS display is panning mode.
# Check the clone mouse pointer is in sync with primary display
# mouse pointer.
#
name = "Enhanced clone mode -- 5 CRT+Ext.LVDS"
general {
displayconfig = 2
portorder = "5340"
clonewidth = 1024
cloneheight = 768
clonerefresh = 60
portdrivers = "ns2501"
}

# set FPINFO so that external LVDS modes are limited to that size.
port 3 {
fpinfo {
width = 1024
height = 768
}
}
}
config 6 {
#
# 855 + SII164 + CRT + DVI
# Choose a CRT that supports bigger than 1024x768 resolution.
# 1. Set 600x800 on CRT and check 600x800 image appeared
# at top,left corner on DVI panel. Or Image should be scaled to
# match the panel size based on DVI panel used.
# 2. Set 1024x1280 on CRT and check DVI display is panning mode.
# Check the clone mouse pointer is in sync with primary display
# mouse pointer.
#
name = "Enhanced clone mode -- 3 CRT+DVI"
general {
displayconfig = 2

```

```
        portorder = "5340"
        clonewidth = 1024
        cloneheight = 768
        clonerefresh = 60
        portdrivers = "sii164"
    }

    # set FPINFO so that DVI modes are limited to that size.
    port 5 {
        general {
            rotation = 90
        }
    }

    port 3 {
        general {
            rotation = 90
        }
        fpinfo {
            width = 1024
            height = 768
        }
    }
}
```

VBIOS

4

4.1 Overview

The Intel Embedded Video BIOS at Version 5.0 incorporates many of the features and capabilities of the Intel® Embedded Graphics Drivers. The legacy VBIOS is still supported and is discussed in [Chapter 5, “Legacy VBIOS.”](#) The 5.0 version of the VBIOS includes the following new features:

- Support of additional DVO transmitters
- A simplified User Build System (UBS)
- Expanded support for dual display configurations and display detection

4.2 System Requirements

The new Video BIOS can be built on a host Microsoft Windows* system and moved to the target system. The host system must have a 32-bit Microsoft Windows* operating system installed with the capability to execute DOS commands from a command line window.

The target system must contain one of the following Intel chipsets:

- Intel® 945G chipset
- Intel® 945GM chipset
- Intel® 915GV chipset
- Intel® 915GM chipset
- Intel® 855GME chipset
- Intel® 852GM chipset
- Intel® 852GME chipset
- Intel® 845GV chipset

The target system must contain a minimum of 64MB of RAM.

4.3 Building the VBIOS with the User Build System (UBS)

The Intel® Embedded VBIOS must be built using command line utilities. The VBIOS is built based on settings in the .pcf file (see [Section 3.5, “Parameter Configuration Format” on page 31](#)). This system of configuration files and command line utilities is called the User Build System (UBS).

The Version 5.0 release packages contain default builds of the TSR and Option ROM for the Intel® 845GV, Intel® 852GM, Intel® 855GME, Intel® 915GV, Intel® 915GME, Intel® 945G, and Intel® 945GM chipsets. The filenames are `TSR-def.exe` and `vga-def.bin`.

The following sections describe how to build a VBIOS using UBS.

4.3.1 Unzipping the VBIOS Zip File

The VBIOS installation is a standard zipped file that should be uncompressed in a folder on the host platform on which you build the VBIOS. When you unzip the VBIOS file, note the following:

- The DOS* executable files used to build the VBIOS may not run properly if the package is extracted in a location with a long directory path. To ensure proper operation, it is advised that the files be placed as close to the root directory of your drive as possible.
- The execution of the UBS utilities require relative locations of the subdirectories. Do not alter the location or names of the subdirectories once the installation file is unzipped.
- In order to build the Option ROM or TSR file, the files in the `... \driver\845` or `... \driver` folder must not be set to read-only. If any of the files involved in the build process are set to read-only, the VBIOS build process fails.

4.3.2 Prerequisites

Before building your VBIOS, you must set up your DOS environment with two items.

1. Install the Open Watcom* C/C++ compiler.

The User Build System for the VBIOS relies on the Open Watcom C/C++ compiler to be able to build a 16-bit DOS binary required for the BIOS. The VBIOS has been tested with version 1.2 of the Open Watcom compiler. This can be downloaded from the following location:

<http://www.openwatcom.com>

2. Set up directory paths.

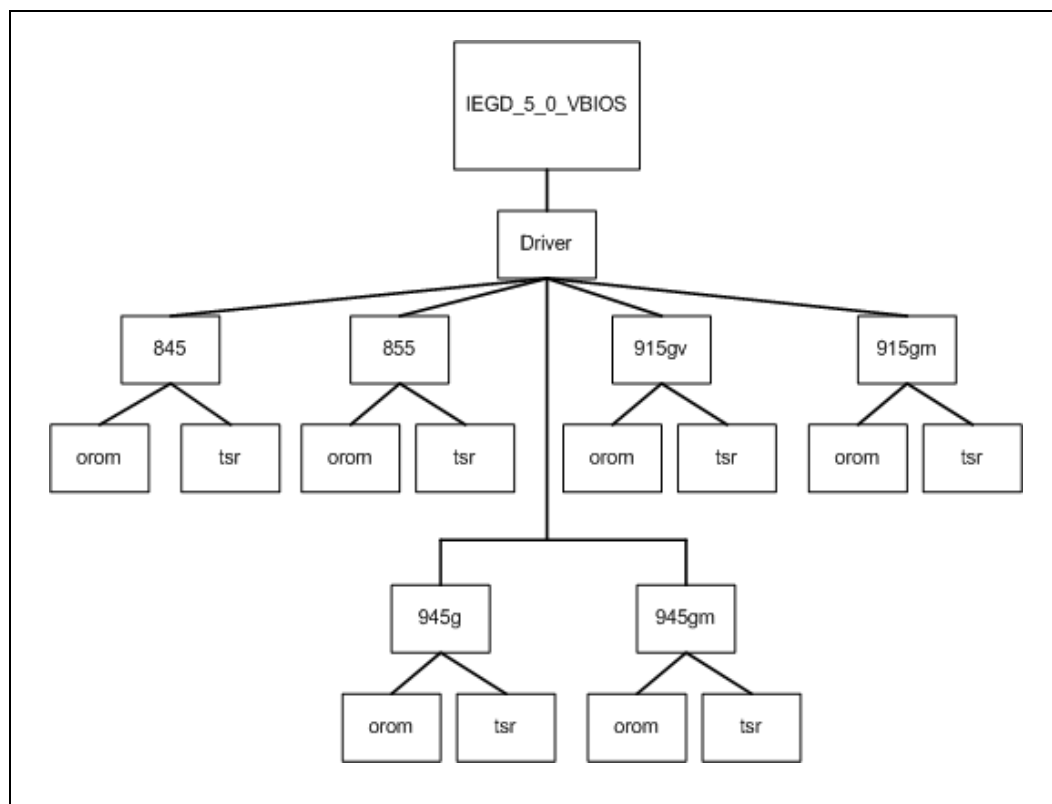
You must set up the PATH environment variable in DOS to be able to execute the Watcom compiler.

4.3.3 Selecting the Build Folder

The 5.0 Version of the VBIOS contains specific folders used for creating a VBIOS that is either an option ROM (OROM) that can be merged with the system BIOS, or an executable Terminate and Stay Resident (TSR) program for debugging purposes. There are also separate directories for the different chipsets that are supported. Depending on which chipset you are using, and whether or not you want an OROM or a TSR, open a DOS command window and move to the appropriate directory. You will execute the build commands in a DOS command window from the selected folder. For example, if you are creating a OROM for a platform with an Intel® 855GME chipset, you would build the VBIOS from the `Driver\855\orom` directory. For building the VBIOS for Intel® 852GME chipset, use the `Driver\855` directories.

Figure 10 shows the directory structure for the Video BIOS.

Figure 10. Video BIOS Directory Structure



4.3.4 Configuring the PCF File for the Video BIOS

This release of the VBIOS now consolidates the configuration into the Parameter Configuration Format file that is also used for configuration of the IEGD. You should create your PCF file according to the instructions in [Section 3.5, “Parameter Configuration Format” on page 31](#).

The display settings will be used in the same manner they are used for the driver. In addition, there is a new `vbios` block that is included in each `config` block of the PCF file with some settings that are specific to the VBIOS. These new settings are described below. Please refer to the `sample.pcf` file in the release package or [Section 3.6, “Sample PCF File” on page 42](#) as an example of these settings.

4.3.4.1 COMMON_TO_PORT

This setting allows you to associate standard display names used in most system BIOSs to specific ports that are recognized by IEGD (e.g., LVDS, DVO-B, DVO-C). The VBIOS makes this association when the VBIOS calls the System BIOS Intel® 5F interrupt functions.

This setting is a six digit number, where each digit is associated with one of the system BIOS displays (from left to right):

```

1 : CRT - Standard analog CRT
2 : TV1 - TV Output 1
3 : EFP1 - DVI Flat Panel 1
4 : LFP - Local Flat Panel (Internal LVDS display)
5 : TV2 - TV Output 2
6 : EFP2 - DVI Flat Panel 2

```

The values above are an example of the typical displays and corresponding order used by a system BIOS. However, this may vary depending on how your system BIOS has implemented the displays and the Intel 5F interrupt functions.

The value in each position in the setting should be the associated port number. Using the typical settings above, if you want to associate CRT in the system BIOS with the internal CRT (port 5) and LFP in the system BIOS with internal LVDS (port 4) in the VBIOS, set `COMMON_TO_PORT` to be 500400.

Warning: This feature must be compatible with the system BIOS. If the system BIOS does not properly implement the Intel 5F functions, then using the `COMMON_TO_PORT` feature could cause unpredictable results with the displays. If you are unsure, set `COMMON_TO_PORT` to all zeros (000000) to disable this feature.

Note: Note that the `displaydetect` parameter must be set to Enabled in order for the `COMMON_TO_PORT` values to be used.

4.3.4.2 `post_display_msg`

This setting is a binary setting that enables (1) or disables (0) post messages to the display.

4.3.4.3 `OEM Vendor Strings`

The following settings are string values that allow you to set the values that are returned from the Intel 4F interrupt functions.

```

oem_string
oem_vendor_name
oem_product_name
oem_product_rev

```

4.3.4.4 `Default Mode Settings`

These settings establish the default VGA or VESA mode to use for the primary (0) and secondary (1) displays. The values should be set to a valid standard VGA or VESA mode (in hexadecimal format, for example, 0x117). Note that a VGA mode can only be set on one display and a second display is disabled unless the `DisplayConfig` parameter is set to `twin` or `clone` mode.

```

default_mode_0
default_mode_1

```

4.3.4.5 Default Refresh Settings

These settings allow you to specify which refresh rate is used for certain VESA modes on the primary and secondary displays. For example, mode 0x117 specifies refresh rates of 60Hz, 75Hz, and 85Hz. This setting allows use to specify which of those three rates to use (specified in decimal, e.g., default_refresh_0=60).

```
default_refresh_0
default_refresh_1
```

4.3.4.6 default_vga_height

This setting allows you to specify which resolution is used for certain VGA modes. Since only one VGA mode can be supported on both displays, this setting applies to the primary display mode (default_mode_0). For example, mode 3 specifies three possible resolutions: 640x200, 640x350, and 720x400. In this example, setting default_vga_height=350 indicates the resolution 640x350.

4.3.5 Building the VBIOS

To build the VBIOS, in a DOS command window run the go.bat batch file from the selected build folder (depending on chipset and if you are building the TSR or Option ROM). The go.bat file takes two command line parameters: the name of the PCF configuration file, and the default configuration ID. The configuration ID should match one of the configuration blocks in the specified PCF file. The configuration ID is optional, and if not specified the first configuration in the PCF file will be used as the default. The command line syntax of the go.bat file is as follows:

```
go <pcf file name> <config ID>
```

Note: In order to build the Option ROM or TSR file, the files in the ... \driver\845 or ... \driver\855 folder must not be set to read-only. If any of the files involved in the build process are read-only, the build process fails.

If you are building a TSR, the resulting file will be TSR.exe. This program can then be copied to the target machine. To run the TSR, boot the target machine with DOS, and then run the TSR.exe from the DOS command line.

If you are building an Option ROM, the resulting file will be VGA.bin. The VGA.bin is the binary option ROM that can be merged with your system BIOS per the instructions provided by your system BIOS vendor.

4.4 VBIOS and Driver Compatibility

4.4.1 Data Dependencies Between VBIOS and Intel Graphics Drivers

The Intel Embedded Graphics Drivers do not depend on any data from the VBIOS, and will either use driver settings or select default values for the attached displays. This allows the driver to properly operate with incompatible BIOS or BIOS replacements.

The Intel Embedded Graphics Drivers will retrieve settings, such as panel ID and other display settings from the Embedded VBIOS. The Embedded VBIOS allows for configuration of display timings that can also be used for the Intel Embedded Graphics Drivers.

4.5 VESA and VGA Video Modes

The VBIOS supports many VESA and standard VGA modes. [Table 13](#) lists the modes and vertical refresh rates that are supported by the VBIOS.

Note: Although IBM* labeled certain EGA modes with a (*) suffix and the VGA modes with a (+) suffix (such as mode 3, 3* and 3+), the VGA modes are so common that this document does not use the (+) suffix to refer to them.

The actual availability of any particular mode depends on the capabilities of the display device, the amount of memory installed, and other system parameters.

Table 13. Supported VGA Video Display Modes (Sheet 1 of 2)

Video Mode	Pixel Resolution	Color Depth (bpp)	Mode Type	Display Adapter	Font Size	Character Resolution	Dot Clock (MHz)	Horiz. Freq. (KHz)	Vert Freq (Hz)	Video Memory (KBytes)
00h	320 x 200	16 (gray) (4 bpp)	Text	CGA	8 x 8	40 x 25	25	31.5	70	256
	320 x 350	16 (gray) (4 bpp)		EGA	8 x 14	40 x 25	25	31.5	70	256
	360 x 400	16 (4 bpp)		VGA	9 x 16	40 x 25	28	31.5	70	256
01h	320 x 200	16 (4 bpp)	Text	CGA	8 x 8	40 x 25	25	31.5	70	256
	320 x 350	16 (4 bpp)		EGA	8 x 14	40 x 25	25	31.5	70	256
	360 x 400	16 (4 bpp)		VGA	9 x 16	40 x 25	28	31.5	70	256
02h	640 x 200	16 (gray) (4 bpp)	Text	CGA	8 x 8	80 x 25	25	31.5	70	256
	640 x 350	16 (gray) (4 bpp)		EGA	8 x 14	80 x 25	25	31.5	70	256
	720 x 400	16 (4 bpp)		VGA	9 x 16	80 x 25	28	31.5	70	256
03h	640 x 200	16 (4 bpp)	Text	CGA	8 x 8	80 x 25	25	31.5	70	256
	640 x 350	16 (4 bpp)		EGA	8 x 14	80 x 25	25	31.5	70	256
	720 x 400	16 (4 bpp)		VGA	9 x 16	80 x 25	28	31.5	70	256
04h	320 x 200	4	Graph	All	8 x 8	40 x 25	25	31.5	70	256
05h	320 x 200	4 (gray)	Graph	CGA	8 x 8	40 x 25	25	31.5	70	256
	320 x 200	4 (gray)		EGA	8 x 8	40 x 25	25	31.5	70	256
	320 x 200	4		VGA	8 x 8	40 x 25	25	31.5	70	256
06h	640 x 200	2	Graph	All	8 x 8	80 x 25	25	31.5	70	256
07h	720 x 350	Mono	Text	MDA	9 x 14	80 x 25	28	31.5	70	256
	720 x 350	Mono		EGA	9 x 14	80 x 25	28	31.5	70	256
	720 x 400	Mono		VGA	9 x 16	80 x 25	28	31.5	70	256
08h-0Ch	Reserved			-		-				

Table 13. Supported VGA Video Display Modes (Continued) (Sheet 2 of 2)

Video Mode	Pixel Resolution	Color Depth (bpp)	Mode Type	Display Adapter	Font Size	Character Resolution	Dot Clock (MHz)	Horiz. Freq. (KHz)	Vert Freq (Hz)	Video Memory (KBytes)
0Dh	320 x 200	16 (4 bpp)	Graph	E/VGA	8 x 8	40 x 25	25	31.5	70	256
0Eh	640 x 200	16 (4 bpp)	Graph	E/VGA	8 x 8	80 x 25	25	31.5	70	256
0Fh	640 x 350	Mono	Graph	E/VGA	8 x 14	80 x 25	25	31.5	70	256
10h	640 x 350	16 (4 bpp)	Graph	E/VGA	8 x 14	80 x 25	25	31.5	70	256
11h	640 x 480	2 (4 bpp)	Graph	VGA	8 x 16	80 x 30	25	31.5	60	256
12h	640 x 480	16 (4 bpp)	Graph	VGA	8 x 16	80 x 30	25	31.5	60	256
13h	320 x 200	256 (8 bpp)	Graph	VGA	8 x 8	40 x 25	25	31.5	70	256

Table 14 lists the VESA modes supported by the Video BIOS.

Table 14. VESA Modes Supported by Video BIOS (Sheet 1 of 3)

Video Mode	Pixel Resolution	Colors (bpp)	Mode Type	Display Adapter	Vertical Frequency (Hz)	Video Memory (MB)
101h	640 x 480	256 (8 bpp)	Graph	VGA	60	0.5
	640 x 480	256 (8 bpp)	Graph	VGA	75	0.5
	640 x 480	256 (8 bpp)	Graph	VGA	85	0.5
103h	800 x 600	256 (8 bpp)	Graph	SVGA	60	1
	800 x 600	256 (8 bpp)	Graph	SVGA	75	1
	800 x 600	256 (8 bpp)	Graph	SVGA	85	1
105h	1024 x 768	256 (8 bpp)	Graph	XVGA	60	1
	1024 x 768	256 (8 bpp)	Graph	XVGA	75	1
	1024 x 768	256 (8 bpp)	Graph	XVGA	85	1

Table 14. VESA Modes Supported by Video BIOS (Continued) (Sheet 2 of 3)

Video Mode	Pixel Resolution	Colors (bpp)	Mode Type	Display Adapter	Vertical Frequency (Hz)	Video Memory (MB)
107h	1280 x 1024	256 (8 bpp)	Graph	SXGA	60	2
	1280 x 1024	256 (8 bpp)	Graph	SXGA	75	2
	1280 x 1024	256 (8 bpp)	Graph	SXGA	85	2
111h	640 x 480	64K (16 bpp)	Graph	VGA	60	1
	640 x 480	64K (16 bpp)	Graph	VGA	75	1
	640 x 480	64K (16 bpp)	Graph	VGA	85	1
114h	800 x 600	64K (16 bpp)	Graph	SVGA	60	2
	800 x 600	64K (16 bpp)	Graph	SVGA	75	2
	800 x 600	64K (16 bpp)	Graph	SVGA	85	2
117h	1024 x 768	64K (16 bpp)	Graph	XVGA	60	2
	1024 x 768	64K (16 bpp)	Graph	XVGA	75	2
	1024 x 768	64K (16 bpp)	Graph	XVGA	85	2
11Ah	1280 x 1024	64K (16 bpp)	Graph	SXGA	60	4
	1280 x 1024	64K (16 bpp)	Graph	SXGA	75	4
	1280 x 1024	64K (16 bpp)	Graph	SXGA	85	4
112	640 x 480	16M (32 bpp)	Graph	VGA	60	2
	640 x 480	16M (32 bpp)	Graph	VGA	75	2
	640 x 480	16M (32 bpp)	Graph	VGA	85	2
115	800 x 600	16M (32 bpp)	Graph	SVGA	60	4
	800 x 600	16M (32 bpp)	Graph	SVGA	75	4
	800 x 600	16M (32 bpp)	Graph	SVGA	85	4

Table 14. VESA Modes Supported by Video BIOS (Continued) (Sheet 3 of 3)

Video Mode	Pixel Resolution	Colors (bpp)	Mode Type	Display Adapter	Vertical Frequency (Hz)	Video Memory (MB)
118	1024 x 768	16M (32 bpp)	Graph	XVGA	60	4
	1024 x 768	16M (32 bpp)	Graph	XVGA	75	4
	1024 x 768	16M (32 bpp)	Graph	XVGA	85	4
11B	1280 x 1024	16M (32 bpp)	Graph	SXGA	60	8
	1280 x 1024	16M (32 bpp)	Graph	SXGA	75	8
	1280 x 1024	16M (32 bpp)	Graph	SXGA	85	8

This page intentionally left blank.

Legacy VBIOS

5

5.1 Overview

In general, a legacy VGA VBIOS provides firmware-based graphics device initialization and rudimentary display output support independent of operating systems, such as text mode support. Operating system independence is accomplished through the legacy IBM® PC method of calling software interrupts, which the system BIOS sets up. Operating systems, prior to display drivers loading, use the rudimentary display output services provided by VBIOS to display data such as informational messages and splash screen. If no display drivers are discovered, operating system and/or applications may rely on the VBIOS during regular operation. In addition, during pre-operating system boot, the system BIOS may also rely on the rudimentary display output services from VBIOS to display items such as informational messages, splash screen, and user setup screen. The Intel Embedded VBIOS provides these rudimentary services by default.

For extended resolutions and industry-standard services, an independent standards organization, the Video Electronics Standards Association (VESA) has defined a set of VBIOS Extensions (VBEs) for the legacy VGA video BIOS. Like the legacy VGA video BIOS interfaces, these extensions are also accessible through the IBM PC method of calling software interrupts. In addition, the latest core standard (VESA/VBE Core 3.0) defines a method for accessing video BIOS functions through 16-bit Protected Mode. The system BIOS, operating systems and/or applications can use these extended resolutions and services. The legacy Intel Embedded VBIOS supports the VESA/VBE Core 3.0 interfaces.

Some information may exist on a platform utilizing the Intel Graphics Controller that is not accessible through any of the standard defined interfaces, such as the legacy VGA Video BIOS interface or the VESA/VBE interfaces. An example of information that cannot be retrieved through any of the standard defined interfaces is what displays are currently attached and detected. Also, some functions may not be accessible through any of the standard interfaces. An example of a function that cannot be accomplished through any standard interface is switching between the displays attached to the Intel graphics controller. For services such as these that are unique to the Intel Graphics Controller hardware, a set of Intel-defined functions exist. These functions are accessible through the legacy IBM PC method of calling software interrupts. Because these Intel functions are not standardized, they can only be used by custom applications or system BIOS.

During platform initialization, the video BIOS may require from the platform some information that it cannot retrieve on its own. On typical platforms, this necessary information is available through the system BIOS. The video BIOS may retrieve the information from system BIOS through Intel-defined System BIOS Hook Functions. For example, if a platform uses a TV as its display, it may use a jumper to indicate NTSC or PAL. The system BIOS can determine the setting of the jumper, then video BIOS will retrieve the setting from system BIOS using the appropriate System BIOS Hook Function. The only purpose for these system BIOS Hooks is to pass information between the Intel Video BIOS and the system BIOS; the functions are not accessible to any external applications. The Intel Embedded VBIOS supports these System BIOS Hook Functions. [Section 5.3.1](#) describes how to turn on and turn off the system BIOS Hook Functions through configuration options.

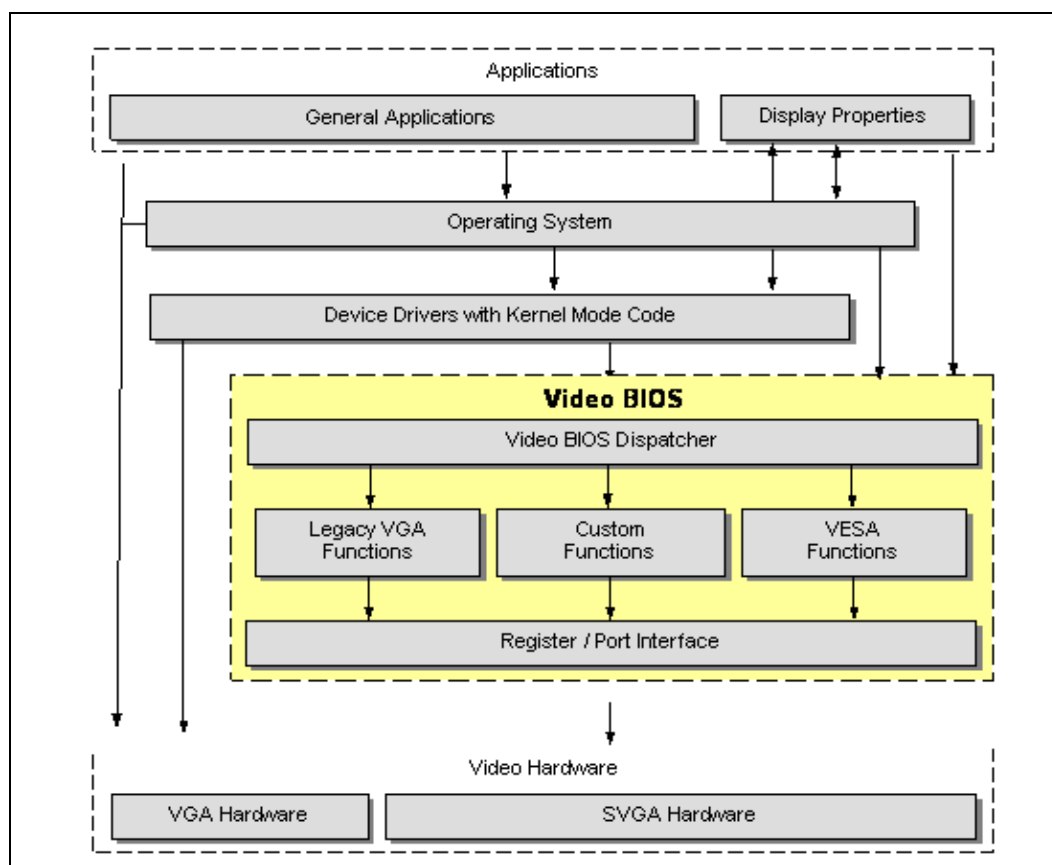
Some of the platform details the VBIOS discovers can be useful for graphics drivers. For example, VBIOS may have already detected attached displays; that information could be passed to the driver so that rediscovery is not necessary. The Intel Embedded VBIOS uses graphics controller registers to pass data to the Intel® Embedded Graphics Drivers. This is intended to be used as VBIOS-to-driver communication only.

Due to the broad range of potential platform configurations, it is not possible to create and provide a single VBIOS binary that can accommodate and automatically detect all different configurations while still fitting within the 64 Kbytes maximum VBIOS size required by customers. Binaries built with basic configurations are already included with the release package, but these work only with specific platform configurations; see the Release Notes in the release package for details. For platforms that do not fall within the default platform definition, the VBIOS requires a configuration effort to accurately reflect the platform configuration. The release package provides a set of tools and configurations that enable creating a new VBIOS binary tailored to the platform. Features such as DVO devices, panel type, panel timings, new mode timings, I²C ports, and more may be configured using the tools provided in the release package. [Section 5.3.1](#) describes how to use the tools and the different configuration options.

5.1.1 VBIOS Model

The VBIOS provides functionality that abstracts the hardware for applications and operating systems. The VBIOS does not actively monitor access to the Intel Graphics Controller hardware registers, nor can it prevent other software from accessing the hardware directly. Often, the VBIOS assumes that it has sole control over the hardware registers; for instance, when a particular mode is set through the VBIOS and another application resets all timing registers for a different mode. Then, the VBIOS will assume that the original mode is still in effect. Therefore, it is highly recommended that applications and System BIOS use the VBIOS functions whenever possible.

Figure 11. VBIOS Model



5.2 Panel Detection

VBIOS accomplishes panel detection through an interface call to system BIOS. This is used to isolate the correct DTD in the table compiled into VBIOS. During boot, VBIOS uses this timing for setting initial graphics modes. Upon transition from system BIOS to OS, the panel ID is transferred to the graphics driver through a semaphore. As with VBIOS, the driver uses the ID to traverse its DTD table located in the registry. The correct DTD is then used to program the panel with the correct display mode.

The VBIOS relies on system BIOS to report the correct panel type associated with the platform. This is conducted via an INT15 system BIOS call by VBIOS. The table below provides the standard panel ID table for three popular DTD panel modes.

Table 15. Example Panel ID Definitions

Panel ID	Mode
1	VGA - 640x480
2	SVGA - 800x600
3	XGA - 1024x768

5.3 Configuration Using User Build System (UBS)

5.3.1 Overview

The VBIOS UBS was developed to:

- Enable users to build a customized VBIOS binary. Customization includes features of the VBIOS, DTD timings, and addition and selection of AIM modules to the present code. This allows flexibility to the end user and on-site compilation to modify the VBIOS binary according to user needs.
- Enable compilation for the Intel® 845 and 85x chipsets.
- Automate modification of assembly sources based on customer input and automatically handle the compilation and linking of selected target object files to produce the output binaries.

The UBS is contained within the `usr_bld` folder that is part of the Intel® Embedded Graphics Drivers VBIOS release package. UBS consists of folders named 845 and 855, representing the supported hardware devices.

The Include folder contains files necessary to create a user-customized object file.

This file is later linked to produce the final binary that is placed in the relevant product folder according to specifications in the VBIOS customization file.

UBS consists of three main components:

- VBIOS launcher
- VBIOS configuration file (`def_eg.txt`)
- VBIOS customization tool

5.3.2 Requirements

- UBS must be installed on a system running a 32-bit Microsoft Windows* operating system with the capability to execute DOS commands from a command line shell.

Note: When extracting the video BIOS package, make sure it is placed as close to the “root” as possible, as there is character limitation in DOS. For example:

Recommended: `C:\IEGD\usr_bld`

Not recommended: `C:\Documents and Settings\Administrator\My Documents\IEGD\usr_bld`

- The target machine must be installed with the open-source Watcom* C/C++ Version 11.0c compiler for DOS and MASM* 6.11 for DOS.
- The paths for the MASM assembler (`ml`) and the Watcom environments must be set up:
 - Go to <Control Panel><System><Advanced><Environment Variables><User Variables>, double click on variable “Path”, and add the full path for `ml.exe` and `nmake.exe` in the <Variable Value> field.

5.3.3 VBIOS Launcher

1. Go to the USR_BLD folder and run go.bat. This is the main execution file.
2. Execute go.bat to create the release.

The go.bat file accepts a configuration file as an input argument. The default configuration file is def_eg.txt. An error is generated if this argument is not specified. This file is explained further in [Section 5.3.4](#)

User-selected configuration options are displayed when this file is executed, and any errors are indicated.

5.3.4 VBIOS Configuration File

The VBIOS configuration file contains customizable user options. A default example file called def_eg.txt is included along with the UBS. This file name has to be a command line parameter for go.bat. Multiple files can be created for different configurations as per user requirements. The file names must be DOS-compliant (maximum of eight characters). The VBIOS configuration file contains many options for user customization. A PCF file is used to configure DTD timings. The file standard.pcf contains an example of timing configurations.

This section explains the available options in the VBIOS Customization Tool (VCT) option configuration file (def_eg.txt). The VCT configuration file is divided into the following categories: VERSION, BUILD, GENERAL, PORT_DEVICE and BOOT. The sequence of these entries matters for correct implementation of the final exe when it is created. Each category has its own collection of options that can be changed. Do not leave any space for entries within a category. These categories are defined in the sections that follow.

5.3.4.1 Version Selection Category

This section is reserved and should not be modified from the default values in the def_eg.txt file distributed with the VBIOS.

5.3.4.2 Build Selection Category

The build selection category is used to select major build options. An example of this is shown in [Figure 12](#) and subsequent sections give a detailed explanation of these options.

Figure 12. Build Settings

[BUILD]	
HARDWARE= 845	; 845,855
VGA=1	; 1 to enable, 0 to disable
VESA=0	; 1 to enable, 0 to disable
INTEL_5F=0	; 1 to enable, 0 to disable
POST_DISPLAY_MSG=1	; 1 to enable, 0 to disable
DISPLAY_STR_MSG="Intel EID Video BIOS"	; not more than 60 chars
VESA_VBE_PM=1	; 1 to enable, 0 to disable

5.3.4.2.1 Hardware

Only two types of hardware input are accepted. Acceptable parameters are:

- 845 for the Intel® 845GV Chipset
- 855 for the Intel® 855GME or the Intel® 852GME chipsets

5.3.4.2.2 VGA

- Enables support for VGA functionality
- Enables selection of legacy VGA interface support
- Selecting VGA support ensures full compatibility with the documented standard IBM* VGA BIOS*, including standard VGA functions, register setting, mode resolutions, and RAM data area values
- Acceptable parameters are:
 - 1 to enable VGA support
 - 0 to disable VGA support

Note: This feature is hard-coded in the VBIOS to always be enabled.

5.3.4.2.3 VESA

- Enables support of VESA functionality
- VESA VBE standard allows software to set non-IBM* standard mode resolutions
- Acceptable parameters are:
 - 1 to enable VESA support
 - 0 to disable VESA support

5.3.4.2.4 INTEL_5F

- BIOS extended interface functions
- Proprietary function calls to control operation of the extended features of the VBIOS
- Acceptable parameters are:
 - 1 to enable Intel_5f support
 - 0 to disable Intel_5f support

Note: Intel_5F settings are not supported in the current release.

5.3.4.2.5 POST_DISPLAY_MSG

- Enable POST display message
- Acceptable parameters are:
 - 1 to enable post display message support
 - 0 to disable post display message support

5.3.4.2.6 **DISPLAY_STR_MSG**

- String to display during POST
- This string is displayed only if POST_DISPLAY_MSG is enabled
- 60-character limit
- Enclose the message with open (“”) and close (”) quotes

Note: This string will only be displayed during POST if POST_DISPLAY_MSG is enabled.

Note: An empty display string message is not advisable (e.g., " ").

5.3.4.2.7 **VESA_VBE_PM**

- Enables or disables power management support through function INT10 4F10.
- Acceptable parameters are:
 - 1 to enable power management support.
 - 0 to disable power management support.

5.3.4.3 **General Selection Category**

The general selection category shows the size and the DOS boot mode. An example is shown in Figure 13.

Figure 13. General Options

```
[GENERAL]
SIZE= 64 ; size in Kbytes
START_BOOT_MODE=0x50 ; DOS boot mode
```

5.3.4.3.1 **Size**

- Allows selection of VBIOS size.
- Acceptable parameter: 64 Kbytes.

5.3.4.3.2 **START_BOOT_MODE**

- Allows selection of required mode when DOS boots up.
- Acceptable parameters for VGA modes are:

Mode	Type	Resolution	Bits per Pixel
0x03	Text only	720x400	4
0x0F	Monotone	640x350	1
0x11	Graphics	640x480	2
0x12	Graphics	640x480	4
0x13	Graphics	320x200	8

Acceptable parameters for VESA modes are:

Mode	Resolution	Colors
0x30	640x480	256
0x32	800x600	256
0x34	1024x768	256
0x40	640x480	32 K
0x41	640x480	64 K
0x42	800x600	32 K
0x43	800x600	64 K
0x44	1024x768	32 K
0x45	1024x768	64 K
0x50	640x480	16 M
0x52	800x600	16 M
0x54	1024x768	16 M

Note: VESA must be enabled under the BUILD category to use the VESA modes.

5.3.4.4 PORT_CONFIG Selection Category

The PORT_CONFIG selection category is used for Port and ADD configuration. An example is shown in [Figure 14](#).

Note: EDID-less DTD timings customization is provided by a separate customization file. The default example is included in the `standard.pcf`. The `standard.pcf` file is parsed by the `pcf2iegd.exe` utility. Once `go.bat` is executed, the DTD timings configuration file is automatically parsed to produce the necessary object file. Customization of DTD timings follow the EDID 1.0 standard. You can make multiple copies of the DTD timings file but must follow the format as shown in `standard.pcf`. The file can be renamed but must be specified in the DTD_TABLE settings in `def_eg.txt`. Only one file can be specified at a time.

Figure 14. Port Configuration Options

```
[PORT_CONFIG]
PORT_DEV_SUPPORT=1           ; 1 to enable, 0 to disable
ADD_CARD_DOWNLOAD=1         ; 1 to enable, 0 to disable
REVERSE_DVO_COLOR_ORDER=0   ; 1 to enable, 0 to disable
PANEL_DETECT=0              ; 1 to enable, 0 to disable
DEFAULT_PANEL_ID=0          ; 0 to disable, any other number = panel ID
DEVICE_SELECT=0             ; 1 to enable, 0 to disable
;DTD_TABLE = "standard.pcf"
;STATIC_DEVICE0=C:\watcom\fal\si154.flx ; user may insert as many as needed
;STATIC_DEVICE1=C:\watcom\falh164.flx   ; in sequence, with no blank lines
;STATIC_DEVICE2=binary_file_name3.xxx   ; in between entries. these client
;STATIC_DEVICE3=binary_file_name4.xxx   ; drivers get statically included
;STATIC_DEVICE4=binary_file_name5.xxx
```

5.3.4.4.1 **PORT_DEV_SUPPORT**

- Allows the selection of AIM device support
- Support for external TV-out digital encoder, DVI devices, and LVDS transmitter.
- If port device support is not selected, then the AIM stub will not be linked in the execution file.
- Acceptable parameters are:
 - 1 to enable port device support
 - 0 to disable port device support

5.3.4.4.2 **ADDCARD_DOWNLOAD**

- This flag dictates whether or not ADD card AIM modules are downloaded and used.
- The parameter is either '1' for TRUE, meaning ADD card download of aim modules is supported, or '0' for FALSE, which means binaries from ADD card AIM modules are not downloaded and run.
- This does not mean that the ADD card DVO device cannot be used at all. Flex-AIM and Static-AIM modules that match the device ID of the ADD card DVO device can still be used.
- The purpose is to enable users to configure the VBIOS to use a flex or statically integrated AIM and not depend on the downloaded AIM module present on an available ADD card.

Note: The ADDCARD_DOWNLOAD setting is not supported in the current release.

5.3.4.4.3 **REVERSE_DVO_COLOR_ORDER**

- This flag allows the user to configure the VBIOS to flip the color order of the DVO signals.
- Causes data signals to be inverted from RGB (MSbs to LSbs) into the opposite direction.
- Parameters are:
 - 0 is default, ensuring normal operation.
 - 1 reverses the color order

5.3.4.4.4 **PANEL_DETECT**

Enables Panel ID detect BIOS function (5F40 through INT 15).

5.3.4.4.5 **DEFAULT_PANEL_ID**

- Assigns default Panel ID to VBIOS if:
 - PANEL_DETECT = 0 or
 - Panel ID detection through system BIOS is not successful.
- Settings DEFAULT_PANEL_ID = 0 means disabling this feature, as in no default panel ID is assigned to VBIOS.

5.3.4.4.6 **DEVICE_SELECT**

- Enables device selection through system BIOS
- Done through INT15h 5F35h.
- Only selected type of devices will be turned on if this feature is enabled.

Note: System BIOS must implement INT15 5F35h to enable user selection to be passed to VBIOS.

5.3.4.4.7 DTD_TABLE

- This entry contains a file name used to configure DTD timings.
- The default file included is standard.pcf
- This file is parsed by the pcf2iegd utility to generate std_dtd.asm, which is then compiled to produce an object file specific to DTD timings. Eventually this file will be automatically linked to the final binary.
- File configuration (standard.pcf) follows standard EDID 1.0 documentation.

5.3.4.4.8 STATIC_DEVICES

- Allows selection of static AIM device support.
- Parameter consists of AIM static binary files.
- Copy these files to the specific intended product folder, or else copy the complete path to identify the files to be linked is required as a parameter.
- With the restriction of VBIOS size, any number of static binaries can be linked.
- Port_dev_support must be enabled for these static files to be linked with the final executable.

5.3.4.5 PORT_DEVICE Selection Category

The PORT_DEVICE selection category is used to configure AIM port device settings. With limitation to VBIOS size, any number of AIM port device settings can be included. Make sure to enable PORT_DEV_SUPPORT in the BUILD category to enable inclusion of these device settings in the final binary.

The configuration of the AIM port device settings dictates whether or not to activate available AIM port modules that are:

- Statically integrated in the VBIOS, called Static-AIM modules. Integration is also done through UBS3.
- Dynamically loaded Flex-AIM modules supported through MBI modules in system BIOS.

When ADDCARD_SUPPORT is enabled, the ADD card configuration always takes highest priority. The ADD card's AIM module is downloaded and activated. If any of the AIM port device settings entries (from this UBS configuration file) match the ADD card AIM settings, they are discarded, and the ADD card AIM settings would be used. The general rules are:

1. If ADDCARD_SUPPORT is TRUE:
 - *And* an ADD card is present, it is downloaded and activated as first priority. Any AIM Port device settings in UBS that conflict with the AIM Port device ID or the ADD card are not used.
 - *And* an ADD card is *not* present, the AIM Port device settings in UBS are used to search for static or Flex-AIM port device modules. Any found that match the device IDs are activated.
2. IF ADDCARD_SUPPORT is FALSE:
 - The AIM Port device settings done in UBS are used to search for static or Flex-AIM port device modules. Any found that match the device IDs are activated.

In any case where the same Flex-AIM and Static-AIMs are available (same because of matching AIM device IDs), VBIOS uses the Static-AIM module.

An example is shown in [Figure 15](#).

Figure 15. Port Device Options

```
[PORT_DEVICE]
Type=Port_EFP_TYPE ; Port device type
ID=SI-164           ; String
Port=Port_DVO_C     ; Port usage = Port_DVO_A, DVO_B, etc
GPIO_BUS=MI2C       ; LTV, DDC, MI2C, etc
I2C_Address=0x70    ; address of i2c device
DDC_BUS=MDVI        ; LTV, DDC, MDVI, etc
Back_Light=1        ; 1 to enable, 0 to disable
```

5.3.4.5.1 Type

- Sets the device type.
- Parameters allowed are:
 - Port_TV_TYPE indicates TV encoder
 - Port_EFP_TYPE indicates TMDS connected to a DFP
 - Port_LFP_TYPE indicates LVDS encoder connected to FP.
 - Port_ADD_EFP_TYPE indicates an additional DFP.

5.3.4.5.2 ID

- Sets the device identification.
- Parameters allowed are string literals with length less than ten characters.
- Parameters are unique strings identifying the AIM module and are provided by the company that created the modules.
- The following table lists known device IDs.

Table 16. Device IDs

Device ID	Product
IN-LVDS	Internal LVDS on Intel® 852GME, Intel® 855GME, Intel® 915GV, and Intel® 915GM chipsets
CH-7009-A	Chrontel* 7009A/7009B/7011 as TV encoder
CH-7009-B	Chrontel* 7009A/7009B/7301 as EFP encoder
NA-2501	National Semiconductor* 2501 as LFP encoder
SI-164	SiliconImage* 154/164 as EFP encoder
TH-164	Thine* 164 as EFP encoder

5.3.4.5.3 Port

- Sets the DVO port for the device.
- Parameters allowed are:
 - Port_DVO_B for all hardware configurations.
 - Port_DVO_C for all hardware configurations.

5.3.4.5.4 GPIO_BUS and DDC_BUS

- General-purpose I/O bus and DDC bus type settings.
- Parameters allowed are:
 - LCKCTRL for general I2C
 - DDCA for analog CRT DDC
 - DDCP for DVI/LVDS DDC
 - MDDC for add card DDC
 - MI2C for add card I2C
 - MDVI for add card I2c or DDC

5.3.4.5.5 I2C_Address

- I2C address.
- Parameter allowed: Hex value.

5.3.4.5.6 BACK_LIGHT

- Enables backlight support for current port through system BIOS.
- Done through INT15h 5F47h.
- Acceptable parameters are:
 - 1 to enable backlight support
 - 0 to disable backlight support

Note: System BIOS must implement INT15 5F47h to enable the backlight for panel selected.

5.3.4.6 BOOT Display Attachment Selection Category

- Allows selection of attach device and boot device
- Delimited by the equal sign; the left side indicates attached device and the right side indicates devices to boot
- Parameters allowed: CRT, TV1, EFP1, LFP, TV2, EFP2
 - TV1 - Indicates TV Out transmitter with single output
 - EFP1 - Indicates DVI transmitter with single output
 - TV2/EFP2 - Indicates transmitter with two external outputs (either TV Out or DVI). For example, Chrontel* CH7009 is a DVI/TV Output Device. Depending on the AIM/PORT client module support, VBIOS can dictate which of these two signals to enable during boot. When TV function is selected, it is designated as TV2; when DVI function is selected, it should be set as EFP2.
- The VBIOS does not allow the combination of TV1 and TV2 to be turned on with a CRT, because display timings for both these display devices are so different that one would cause the other to be corrupted. Thus, the VBIOS only allows TV1 or TV2 to be turned on.
- An example is shown in [Figure 16](#).

Figure 16. Boot Options

```
[Boot]
; Attached Dev Boot Dev
CRT=CRT
TV1=TV1
EFP1=EFP1
LFP,CRT=CRT
LFP,TV1=TV1
LFP,EFP1=EFP1
CRT,TV1=CRT
CRT,EFP1=CRT,EFP1
LFP,CRT,EFP1=CRT,EFP1
```

5.3.5 VBIOS Customization Tool (VCT)

The VCT contains a file called `vct.exe`. This tool is responsible for digesting the user-customizable file described above. It generates files called `config.asm` and `std_dtd.asm` (if the DTD option is selected) into a product-specific folder. The tool compiles this file to generate an object file. VCT then creates the target binary. The binary is located in the output folder of the hardware folder selected according to the requested hardware revision. The binary will be a RAM TSR (`dosvga.exe`) or a VGA VBIOS binary extractor (`romvga.exe`). `VGA.bin` is also automatically generated.

5.3.6 VBIOS Tips

1. During dosvga execution, to enable the static aim/port integration of AIM/PORT client modules, remember to:
 - a. Run the UBS with the correct modifications in the Video Configuration File.
 - b. Run "dosvga xxx.a yyy.b ..." where xxx.a and yyy.b are the aim/port client modules binaries that have been added in the Video Configuration File such as si154.flx and etc.
2. Make sure all the files in the `... \driver \usr_bld` folder are not set to read-only. If any of the files involved in the build process are set to read-only, the build process fails.
3. When UBS completes, if there was static integration of AIM/PORT client modules, the UBS merely runs "romvga xxx.a yyy.b ..." where xxx.a and yyy.b are the aim/port client modules binaries that have been added in the video configuration file such as si154.flx, etc. Thus, if romvga did not find the file in the proper folder, the user can rerun "romvga xxx.a yyy.b ..." and not have to repeat the entire UBS from go.bat. If VGA.BIN is not present, it indicates a failure in finding the specified file in STATIC_DEVICE entry at the VBIOS configuration file.

5.4 System BIOS Interface

The VBIOS is software that resides as an image in a ROM that is accessible by system BIOS during system boot time. The VBIOS image may be integrated with the system BIOS, which resides in the ROM on the motherboard, or the VBIOS may be a stand-alone image that resides in a ROM on a GC add-in board. In either case, the VBIOS image is never run in place, and some interfaces between VBIOS and system BIOS are required to improve flexibility.

5.5 VBIOS and Driver Compatibility

5.5.1 Data Dependencies Between VBIOS and Intel Graphics Drivers

The Intel Embedded Graphics Drivers do not depend on any data from the VBIOS, and will either use driver settings or select default values for the attached displays. This allows the driver to properly operate with incompatible BIOS or BIOS replacements.

The Intel Embedded Graphics Drivers will retrieve settings, such as panel ID and other display settings from the Embedded VBIOS. The Embedded VBIOS allows for configuration of display timings that can also be used for the Intel Embedded Graphics Drivers.

5.6 Video Modes

The VBIOS supports standard VGA modes. [Table 17](#) lists the modes and vertical refresh rates that are supported by the VBIOS.

Note: Although IBM* labeled certain EGA modes with a (*) suffix and the VGA modes with a (+) suffix (such as mode 3, 3* and 3+), the VGA modes are so common that this document does not use the (+) suffix to refer to them.

The actual availability of any particular mode depends on the capabilities of the display device, the amount of memory installed, and other system parameters.

Table 17. Standard VGA Video Display Modes

Video Mode	Pixel Resolution	Color Depth	Mode Type	Display Adapter	Font Size	Character Resolution	Dot Clock (MHz)	Horiz. Freq. (KHz)	Vert Freq (Hz)	Video Memory (Kbytes)
00h	320 x 200	16 (gray)	Text	CGA	8 x 8	40 x 25	25	31.5	70	256
	320 x 350	16 (gray)		EGA	8 x 14	40 x 25	25	31.5	70	256
	360 x 400	16		VGA	9 x 16	40 x 25	28	31.5	70	256
01h	320 x 200	16	Text	CGA	8 x 8	40 x 25	25	31.5	70	256
	320 x 350	16		EGA	8 x 14	40 x 25	25	31.5	70	256
	360 x 400	16		VGA	9 x 16	40 x 25	28	31.5	70	256
02h	640 x 200	16 (gray)	Text	CGA	8 x 8	80 x 25	25	31.5	70	256
	640 x 350	16 (gray)		EGA	8 x 14	80 x 25	25	31.5	70	256
	720 x 400	16		VGA	9 x 16	80 x 25	28	31.5	70	256
03h	640 x 200	16	Text	CGA	8 x 8	80 x 25	25	31.5	70	256
	640 x 350	16		EGA	8 x 14	80 x 25	25	31.5	70	256
	720 x 400	16		VGA	9 x 16	80 x 25	28	31.5	70	256
04h	320 x 200	4	Graph	All	8 x 8	40 x 25	25	31.5	70	256
05h	320 x 200	4 (gray)	Graph	CGA	8 x 8	40 x 25	25	31.5	70	256
	320 x 200	4 (gray)		EGA	8 x 8	40 x 25	25	31.5	70	256
	320 x 200	4		VGA	8 x 8	40 x 25	25	31.5	70	256
06h	640 x 200	2	Graph	All	8 x 8	80 x 25	25	31.5	70	256
07h	720 x 350	Mono	Text	MDA	9 x 14	80 x 25	28	31.5	70	256
	720 x 350	Mono		EGA	9 x 14	80 x 25	28	31.5	70	256
	720 x 400	Mono		VGA	9 x 16	80 x 25	28	31.5	70	256
08h-0Ch	Reserved			-		-				
0Dh	320 x 200	16	Graph	E/VGA	8 x 8	40 x 25	25	31.5	70	256
0Eh	640 x 200	16	Graph	E/VGA	8 x 8	80 x 25	25	31.5	70	256
0Fh	640 x 350	Mono	Graph	E/VGA	8 x 14	80 x 25	25	31.5	70	256
10h	640 x 350	16	Graph	E/VGA	8 x 14	80 x 25	25	31.5	70	256
11h	640 x 480	2	Graph	VGA	8 x 16	80 x 30	25	31.5	60	256
12h	640 x 480	16	Graph	VGA	8 x 16	80 x 30	25	31.5	60	256
13h	320 x 200	256	Graph	VGA	8 x 8	40 x 25	25	31.5	70	256

Table 18 lists VESA modes supported by the Legacy VBIOS

Table 18. VESA Modes Supported by Legacy VBIOS

Mode	VBE Mode	Resolution and bpp (bits per pixel)	Colors	Type	Dot Clock (MHz)	Frequency (Hz)	Video Memory (MB)
30h	101h	640x480x8	256	Graph	25.175	60	0.29
32h	103h	800x600x8	256	Graph	40	60	0.46
34h	105h	1024x768x8	256	Graph	65	60	0.75
38h	107h	1280x1024x8	256	Graph	108	60	1.25
3Ah	13Ah	1600x1200x8	256	Graph	162	60	1.83
3Ch	13Ch	1920x1440x8	256	Graph	234	60	2.77
41h	111h	640x480x16	64K	Graph	25.175	60	0.59
43h	114h	800x600x16	64K	Graph	40	60	0.92
45h	117h	1024x768x16	64K	Graph	65	60	1.50
49h	11Ah	1280x1024x16	64K	Graph	108	60	2.5
4Bh	14Bh	1600x1200x16	64K	Graph	162	60	3.67
4Dh	14Dh	1920x1440x16	64K	Graph	234	60	5.5
50h	112h	640x480x32	16M	Graph	25.175	60	1.17
52h	115h	800x600x32	16M	Graph	40	60	1.83
54h	118h	1024x768x32	16M	Graph	65	60	3.0
58h	11Bh	1280x1024x32	16M	Graph	108	60	5.0
5Ah	15Ah	1600x1200x32	16M	Graph	162	60	7.32

Configuring and Installing Microsoft Windows* Drivers

6

6.1 Overview

This section describes the driver-level information for the Microsoft Windows* operating system, which includes the following¹:

- Microsoft Windows* 2000
- Microsoft Windows XP*
- Microsoft Windows XP Embedded*

Note: Configuration and Installation information for the Microsoft Windows CE* .NET operating system is described in [Chapter 7, “Installing and Configuring Microsoft Windows CE* Drivers”](#).

6.2 Configuration Information

6.2.1 Universal INF Configuration

Multiple display configurations can be specified in a single INF file. Each configuration is uniquely identified by the ConfigId parameter.

The driver reads the PanelId from the System BIOS during initialization and uses the configuration whose ConfigId matches the PanelId. If the System BIOS does not set a valid PanelId (for example, panelId = 0), the driver reads a configuration using ConfigId = 1. (A ConfigId value of 0 is invalid.)

You can override this behavior by specifying a ConfigId parameter as follows:

```
HKR,, ConfigId, %REG_DWORD%, %DEFAULT_CONFIG_ID%
```

In this case, the driver ignores the PanelId returned by the System BIOS. Instead, the IEGD uses the configuration information using the specified ConfigId.

6.2.2 INF File Backward Compatibility

Version 5.0 of the IEGD contains a new INF file format. You cannot use the new INF file with pre-5.0 versions of the IEGD. However, you can still use pre-5.0 INF file formats with version 5.0 of the IEGD.

1. These versions of the drivers are not WHQL (Windows Hardware Quality Labs) certified.

6.2.2.1 INF File Backward Compatibility with IEGD Version 4.0

Version 4.0 of the IEGD provides backward compatibility with pre-4.0 versions of the INF file. This support is implemented through the `PcfVersion` key in the INF file, shown below:

```
HKR,, PcfVersion, %REG_DWORD%, 0x0400
```

The IEGD uses this key to determine which version of the .inf file it is interpreting. When this key is present in the .inf file and its value is 0x0400, the driver reads it as a 4.0 .inf file. If this key is omitted from the .inf file or if its value is less than 0x0400, the driver reads the .inf file as a pre-4.0 file.

Note the following rules:

- If you use a pre-4.0 version of the .inf file with version 4.0 of the IEGD, the driver translates pre-4.0 configuration parameters to 4.0 parameters.
- You cannot use 4.0 parameters in a pre-4.0 .inf file. If you try, the driver ignores them.
- You cannot use pre-4.0 parameters in a 4.0 .inf file. If you try, the driver ignores them.

For example, the `usestdtimings` parameter is a pre-4.0 parameter. If it is specified in a 4.0 INF file, the driver ignores it. Similarly, if you attempt to add the `edid_avail` and `edid_non_avail` parameters to a pre-4.0 .inf file (that is, an .inf file where the `PcfVersion` key is not present), they are ignored by the driver.

The `PcfVersion` key is generated automatically by the `pcf2iegd -inf` utility and is placed in the `[ssigd_SoftwareDeviceSettings]` section of the .inf file. You cannot add it to the PCF file and there is usually no need to add it yourself to the .inf file. The default `ssigd.inf` file for version 5.0 already contains the `PcfVersion` key. Refer to [Appendix B, “Example INF File”](#) to view a sample 5.0 .inf file.

6.2.3 Dual Panel Configuration

Below are the settings required to set the INF file to enable extended display configurations. Typically, these settings are output from the `pcf2iegd.exe` utility using the `-inf` option. However, the INF file may also be edited directly. See [Table 19](#) for a description of these settings.

```
HKR, Config\%DEFAULT_CONFIG_ID%\General, DisplayConfig, %REG_DWORD%, 8
HKR, Config\%DEFAULT_CONFIG_ID%\General, PortOrder, %REG_SZ%, "5200"
```

6.2.4 Intel® 855GME Dual Display Example

The table below presents the dual display example for the Intel® 855GME chipset. For other chipsets, refer to [Section 5.4.2](#) to determine available output and ports.

Table 19. Example of Intel® 855GME Dual Display Parameter Setting (Sheet 1 of 2)

Dual Display Combination	Port Order
CRT + Internal LVDS	"5400"
CRT + DVOB	"5200"
CRT + DVOC	"5300"
Internal LVDS + CRT	"4500"
Internal LVDS + DVOB	"4200"

Table 19. Example of Intel® 855GME Dual Display Parameter Setting (Sheet 2 of 2)

Internal LVDS + DVOG	"4300"
DVOB + CRT	"2500"
DVOB + Internal LVDS	"2400"
DVOB + DVOG	"2300"
DVOG + CRT	"3500"
DVOG + Internal LVDS	"3400"
DVOG + DVOB	"3200"

6.2.5 Creating Registry Settings for Graphics Driver INF file

As described in [Section 6.2.3](#), the driver settings can be configured with a PCF file, and the `pcf2iegd.exe` utility can be used to convert the PCF settings to INF settings. For example,

```
pcf2iegd -inf system.pcf > system.inf
```

This command generates the following output, which is then inserted into the graphics driver INF file before driver installation. The `pcf2iegd.exe` tool simply translates the format of the PCF file to the INF file. See [Table 9, "Parameter Configuration Format" on page 33](#) for details on the specific PCF settings and values, which also apply to the settings and values of the INF file. The values of the INF file may also be directly modified. See the example below for syntax and usage. Also, see [Appendix B, "Example INF File"](#) for a complete sample INF file.

```
HKR,,                                PcfVersion, REG_DWORD, 0x0400
HKR, pd                             ,          siil64, %REG_DWORD%, 1
HKR, pd                             ,          ch7009, %REG_DWORD%, 0
HKR, pd                             ,          ns2501, %REG_DWORD%, 0
HKR, pd                             ,          th164, %REG_DWORD%, 0
HKR, pd                             ,          fs454, %REG_DWORD%, 0
HKR, pd                             ,          ns387, %REG_DWORD%, 0
HKR, pd                             ,          lvds, %REG_DWORD%, 0
HKR, pd                             ,          ch7017, %REG_DWORD%, 0
HKR, Config\1                        ,          name, %REG_SZ%, "Intel(R) Embedded
Graphics Driver Platform Configuration File"
HKR, Config\1\General                , DisplayConfig, %REG_DWORD%, 4
HKR, Config\1\General                , DisplayDetect, %REG_DWORD%, 1
HKR, Config\1\General                , PortOrder, %REG_SZ%, "3520"
HKR, Config\1\Port\3                ,          name, %REG_SZ%, "port3"
HKR, Config\1\Port\3\General         , Rotation, %REG_DWORD%, 0
HKR, Config\1\Port\3\General         , Edid, %REG_DWORD%, 1
HKR, Config\1\Port\3\General         , EdidAvail, %REG_DWORD%, 3
HKR, Config\1\Port\3\General         , EdidNotAvail, %REG_DWORD%, 1
HKR, Config\1\Port\3\General         , MultiDvo, %REG_DWORD%, 1
HKR, Config\1\Port\3\Dvo             , DdcPin, %REG_DWORD%, 3
HKR, Config\1\Port\3\Dvo             , DdcSpeed, %REG_DWORD%, 100
HKR, Config\1\Port\3\Dvo             , ddcdab, %REG_DWORD%, 165
HKR, Config\1\Port\3\Dvo             , I2cPin, %REG_DWORD%, 4
HKR, Config\1\Port\3\Dvo             , I2cSpeed, %REG_DWORD%, 120
HKR, Config\1\Port\3\Dvo             , I2cDab, %REG_DWORD%, 104
HKR, Config\1\Port\3\FpInfo          , Width, %REG_DWORD%, 800
HKR, Config\1\Port\3\FpInfo          , Height, %REG_DWORD%, 600
HKR, Config\1\Port\3\FpInfo          , bklmethod, %REG_DWORD%, 1
HKR, Config\1\Port\3\FpInfo          , BkltT1, %REG_DWORD%, 50
HKR, Config\1\Port\3\FpInfo          , BkltT2, %REG_DWORD%, 25
HKR, Config\1\Port\3\FpInfo          , BkltT3, %REG_DWORD%, 15
HKR, Config\1\Port\3\FpInfo          , BkltT4, %REG_DWORD%, 10
HKR, Config\1\Port\3\FpInfo          , BkltT5, %REG_DWORD%, 5
HKR, Config\1\Port\3\FpInfo          , GpioPinVdd, %REG_DWORD%, 3
HKR, Config\1\Port\3\FpInfo          , GpioPinVee, %REG_DWORD%, 5
HKR, Config\1\Port\3\FpInfo          , gpiopinenable, %REG_DWORD%, 1
```

```

HKR, Config\1\Port\3\Dtd\1, PixelClock, %REG_DWORD%, 30000
HKR, Config\1\Port\3\Dtd\1, HorzActive, %REG_DWORD%, 800
HKR, Config\1\Port\3\Dtd\1, HorzSync, %REG_DWORD%, 32
HKR, Config\1\Port\3\Dtd\1, HorzSyncPulse, %REG_DWORD%, 120
HKR, Config\1\Port\3\Dtd\1, HorzBorder, %REG_DWORD%, 0
HKR, Config\1\Port\3\Dtd\1, HorzBlank, %REG_DWORD%, 224
HKR, Config\1\Port\3\Dtd\1, VertActive, %REG_DWORD%, 600
HKR, Config\1\Port\3\Dtd\1, VertSync, %REG_DWORD%, 3
HKR, Config\1\Port\3\Dtd\1, VertSyncPulse, %REG_DWORD%, 2
HKR, Config\1\Port\3\Dtd\1, VertBorder, %REG_DWORD%, 0
HKR, Config\1\Port\3\Dtd\1, VertBlank, %REG_DWORD%, 24
HKR, Config\1\Port\3\Dtd\1, HorzSize, %REG_DWORD%, 25
HKR, Config\1\Port\3\Dtd\1, VertSize, %REG_DWORD%, 19
HKR, Config\1\Port\3\Dtd\1, Flags, %REG_DWORD%, 0
HKR, Config\1\Port\3\Attr, 2, %REG_DWORD%, 50
HKR, Config\1\Port\3\Attr, 4, %REG_DWORD%, 25
HKR, Config\1\Port\3\Attr, 6, %REG_DWORD%, 75
HKR, Config\1\Port\3\Attr, 8, %REG_DWORD%, 100
HKR, Config\1\Port\3\Attr, 16, %REG_DWORD%, 35
HKR, Config\1\Port\3\Attr, 9, %REG_DWORD%, 45
HKR, Config\2, name, %REG_SZ%, "Intel(R) Embedded Graphics
Driver Platform Configuration File"
HKR, Config\2\General, DisplayConfig, %REG_DWORD%, 2
HKR, Config\2\General, DisplayDetect, %REG_DWORD%, 1
HKR, Config\2\General, PortOrder, %REG_SZ%, "3520"
HKR, Config\2\Port\3, name, %REG_SZ%, "port3"
HKR, Config\2\Port\3\General, Rotation, %REG_DWORD%, 0
HKR, Config\2\Port\3\General, Edid, %REG_DWORD%, 1
HKR, Config\2\Port\3\General, EdidAvail, %REG_DWORD%, 3
HKR, Config\2\Port\3\General, EdidNotAvail, %REG_DWORD%, 1
HKR, Config\2\Port\3\General, MultiDvo, %REG_DWORD%, 1
HKR, Config\2\Port\3\Dvo, DdcPin, %REG_DWORD%, 3
HKR, Config\2\Port\3\Dvo, DdcSpeed, %REG_DWORD%, 100
HKR, Config\2\Port\3\Dvo, ddc dab, %REG_DWORD%, 165
HKR, Config\2\Port\3\Dvo, I2cPin, %REG_DWORD%, 4
HKR, Config\2\Port\3\Dvo, I2cSpeed, %REG_DWORD%, 120
HKR, Config\2\Port\3\Dvo, I2cDab, %REG_DWORD%, 104
HKR, Config\2\Port\3\FpInfo, Width, %REG_DWORD%, 800
HKR, Config\2\Port\3\FpInfo, Height, %REG_DWORD%, 600
HKR, Config\2\Port\3\FpInfo, bklmethod, %REG_DWORD%, 1
HKR, Config\2\Port\3\FpInfo, BkltT1, %REG_DWORD%, 50
HKR, Config\2\Port\3\FpInfo, BkltT2, %REG_DWORD%, 25
HKR, Config\2\Port\3\FpInfo, BkltT3, %REG_DWORD%, 15
HKR, Config\2\Port\3\FpInfo, BkltT4, %REG_DWORD%, 10
HKR, Config\2\Port\3\FpInfo, BkltT5, %REG_DWORD%, 5
HKR, Config\2\Port\3\FpInfo, GpioPinVdd, %REG_DWORD%, 3
HKR, Config\2\Port\3\FpInfo, GpioPinVee, %REG_DWORD%, 5
HKR, Config\2\Port\3\FpInfo, gpiopinenable, %REG_DWORD%, 1
HKR, Config\2\Port\3\Dtd\2, PixelClock, %REG_DWORD%, 25000
HKR, Config\2\Port\3\Dtd\2, HorzActive, %REG_DWORD%, 640
HKR, Config\2\Port\3\Dtd\2, HorzSync, %REG_DWORD%, 8
HKR, Config\2\Port\3\Dtd\2, HorzSyncPulse, %REG_DWORD%, 128
HKR, Config\2\Port\3\Dtd\2, HorzBorder, %REG_DWORD%, 0
HKR, Config\2\Port\3\Dtd\2, HorzBlank, %REG_DWORD%, 160
HKR, Config\2\Port\3\Dtd\2, VertActive, %REG_DWORD%, 480
HKR, Config\2\Port\3\Dtd\2, VertSync, %REG_DWORD%, 10
HKR, Config\2\Port\3\Dtd\2, VertSyncPulse, %REG_DWORD%, 2
HKR, Config\2\Port\3\Dtd\2, VertBorder, %REG_DWORD%, 0
HKR, Config\2\Port\3\Dtd\2, VertBlank, %REG_DWORD%, 45
HKR, Config\2\Port\3\Dtd\2, HorzSize, %REG_DWORD%, 21
HKR, Config\2\Port\3\Dtd\2, VertSize, %REG_DWORD%, 16
HKR, Config\2\Port\3\Dtd\2, Flags, %REG_DWORD%, 0
HKR, Config\2\Port\3\Attr, 2, %REG_DWORD%, 50
HKR, Config\2\Port\3\Attr, 4, %REG_DWORD%, 25
HKR, Config\2\Port\3\Attr, 8, %REG_DWORD%, 100
HKR, Config\2\Port\3\Attr, 16, %REG_DWORD%, 35

```


6.2.6 Dynamic Port Driver Configuration

The IEGD supports many third-party digital transmitters connected to the DVO ports of the GMCH through device drivers called port drivers. These port drivers are dynamically loaded at startup. The driver configuration can be modified to add or remove availability of specific port drivers.

This section describes the portions of the `ssigd.inf` file that can be modified to either add or remove a port driver for the Microsoft Windows* version of the Intel® Embedded Graphics Drivers.

Note: The `pcf2iegd.exe` utility does not generate settings for the dynamic port drivers feature, so any changes must be made directly to the `ssigd.inf` file as specified in this section.

Also note that the `[ssigd_SoftwareDeviceSettings]` block is used for the i81x, i845, and i85x configurations while the `[ssigd_gdg_SoftwareDeviceSettings]` block is used for 915x configurations. Output from the `pcf2iegd` utility must be placed in the appropriate section for these chipset configurations.

6.2.6.1 DestinationDirs

The first step in either adding or removing a port driver is to identify where the driver can locate the port driver. This is done in the `[DestinationDirs]` block of the `.inf` file. Below are the default settings in the `ssigd.inf` file:

```
[DestinationDirs]
DefaultDestDir      = 11
ssigd.Display        = 11
ssigd.Miniport       = 12
ssigd.ch7009         = 12
ssigd.fs454          = 12
ssigd.lvds           = 12
ssigd.ns2501         = 12
ssigd.ns387          = 12
ssigd.sii164         = 12
ssigd.th164          = 12
ssigd.sdvo           = 12
```

The possible locations of a port driver and the associated values of these locations are as follows:

- 10 = Windows directory
- 11 = System directory
- 12 = Drivers directory
- 17 = INF directory
- 20 = Help directory

To remove one or more port drivers, delete the associated line from the `DestinationDirs` block. To add a port driver, add the associated line into the `DestinationDirs` block. For example, to add a port driver for a Chronitel* CH7017 device, add the following line:

```
ssigd.ch7017        = 12
```

6.2.6.2 SourceDisksFiles

The next step to either add or remove a port driver is to identify the specific port driver file names in the SourceDisksFiles blocks. There are two blocks, and the default settings are as follows:

```
[SourceDisksFiles.x86]
igdmini.sys = 1
igddis.dll  = 1
ch7009.sys  = 1
fs454.sys   = 1
lvds.sys    = 1
ns2501.sys  = 1
ns387.sys   = 1
sii164.sys  = 1
th164.sys   = 1
sdvo.sys    = 1

[SourceDisksFiles.ia64]
igdmini.sys = 1
igddis.dll  = 1
ch7009.sys  = 1
fs454.sys   = 1
lvds.sys    = 1
ns2501.sys  = 1
ns387.sys   = 1
sii164.sys  = 1
th164.sys   = 1
sdvo.sys    = 1
```

To remove a port driver, delete the associated lines in both the [SourceDisksFiles.x86] and [SourceDisksFiles.ia64] blocks. To add a port driver, add the associated line to both blocks. For example, to add a port driver for a Chrontel* CH7017 device that is named ch7017.sys, add the following line to both blocks:

```
ch7017.sys = 1
```

6.2.6.3 Name and File Association

Next, associate the name of the port driver defined in the DestinationDirs and DefaultInstall sections with the specific file name used in the SourceDiskFiles section. Below are the default settings:

```
[ssigd.ch7009]
ch7009.sys

[ssigd.fs454]
fs454.sys

[ssigd.lvds]
lvds.sys

[ssigd.ns2501]
ns2501.sys

[ssigd.ns387]
ns387.sys

[ssigd.sii164]
```

```
sii164.sys

[ssigd.th164]
th164.sys

[ssigd.sdvo]
sdvo.sys
```

To remove a port driver, remove the name and filename association block. To add a port driver, add a new block that associates the name and filename. Continuing the Chrontel* 7017 example, the following lines would be added to the `ssigd.inf` file:

```
[ssigd.ch7017]
ch7017.sys
```

6.2.6.4 PortDrivers Registry Key

The next step is to modify the registry key in the `[ssigd_SoftwareDeviceSettings]` section that defines the list of available port drivers. Below is the default value of this registry key in the `ssigd.inf` file:

```
HKR,, PortDrivers, %REG_SZ%, "ch7009 fs454 lvds ns2501 ns387 sii164
th164"
```

Remove or add port driver names as appropriate to the list of port drivers specified within the quoted string. For example, to add support for Chrontel* 7017, the registry key would be defined as follows:

```
HKR,, PortDrivers, %REG_SZ%, "ch7009 fs454 lvds ns2501 ns387 sii164
th164 ch7017"
```

6.2.7 Creating an .sld file for Microsoft Windows XP Embedded* Systems

Microsoft Windows XP Embedded* operating systems require the use of an .sld (system level definitions) file. A default .sld file is provided. However, if you customize the .inf file, you must use it to create a new .sld file.

Once you have customized the .inf file, use the following procedure to create the .sld file.

1. Comment out the sections for the GMCH family of chipsets that you are not using. For example, if you want to create an .sld file for the alm family, comment out the `ssigd_wht` and `ssigd_nap` sections in the customized .inf file by adding a semicolon at the beginning of each line as shown below. Also comment out the Reboot line in the `ssigd_alm` section.

```
; [DefaultInstall]
; CopyFiles = ssigd.Miniport, ssigd.Display, ssigd.ch7009, ...
; Reboot

[ssigd_alm]
CopyFiles = ssigd.Miniport, ssigd.Display, ssigd.D3D_alm, ...
; Reboot

; [ssigd_wht]
; CopyFiles = ssigd.Miniport, ssigd.Display, ssigd.D3D_wht, ...
; Reboot
```

```
;[ssigd_nap]
;CopyFiles = ssigd.Miniport, ssigd.Display, ssigd.D3D_gdg,
;Reboot
```

2. Save the .inf file and run Component Designer.
3. In the **File** menu, select **Import**.
4. In the **Choose File for Import** dialog, select **Setup Information files (*.inf)** in the **File of type** listbox.
5. Browse to the directory that contains the saved .inf file and select it.
6. In the **Inf Processing Options** dialog, select **Automatic** in the **Parsing Options** dialog.
7. Click the **Start** button in the **Import File** dialog. This creates the .sld file.
8. Check for errors in the log window. If there are no errors, **Save** the .sld file.

To create an .sld file for the Intel® 815/E family, comment out the `ssigd_alm` and `ssigd_nap` section and repeat these steps.

6.2.8 Changing Default Display Mode

When installing the Intel® Embedded Graphics Drivers, Microsoft Windows* selects a default display mode for the initial startup of the system. This is a 640 x 480 resolution in four-bit color mode.

In some cases, particularly with EDID-less LVDS displays, the 640 x 480 resolution may not be supported, so the default mode selected by Microsoft Windows* must be changed. Otherwise, the display may not work after installation.

This default mode can be changed by adding the following registry keys to the `[ssigd_SoftwareDeviceSettings]` section of the `ssigd.inf` file:

```
HKR,, DefaultSettings.XResolution, %REG_DWORD%, 1024
HKR,, DefaultSettings.YResolution, %REG_DWORD%, 768
HKR,, DefaultSettings.BitsPerPel, %REG_DWORD%, 32
HKR,, DefaultSettings.VRefresh, %REG_DWORD%, 60
```

The example above makes the default resolution 1024 x 768, with a 32-bit color depth and a refresh rate of 60MHz.

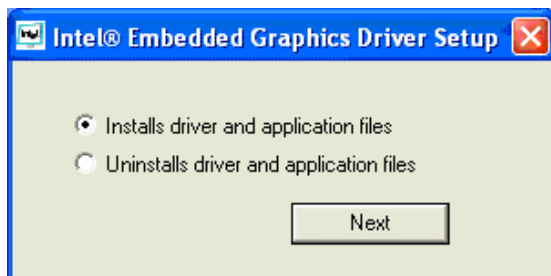
6.3 Installing the IEGD on Microsoft Windows*

You can install and uninstall the IEGD on a Microsoft Windows* system by using the `setup.exe` program located in the `IEGD_5_0_Windows\utilites` folder. (Use the `setupNT.exe` utility to install the driver on a Microsoft Windows NT* system.) The following procedure shows how to install the IEGD. [Section 6.4, “Uninstalling Version 5.0 of the Driver” on page 94](#) provides instructions for uninstalling the 5.0 version of the IEGD.

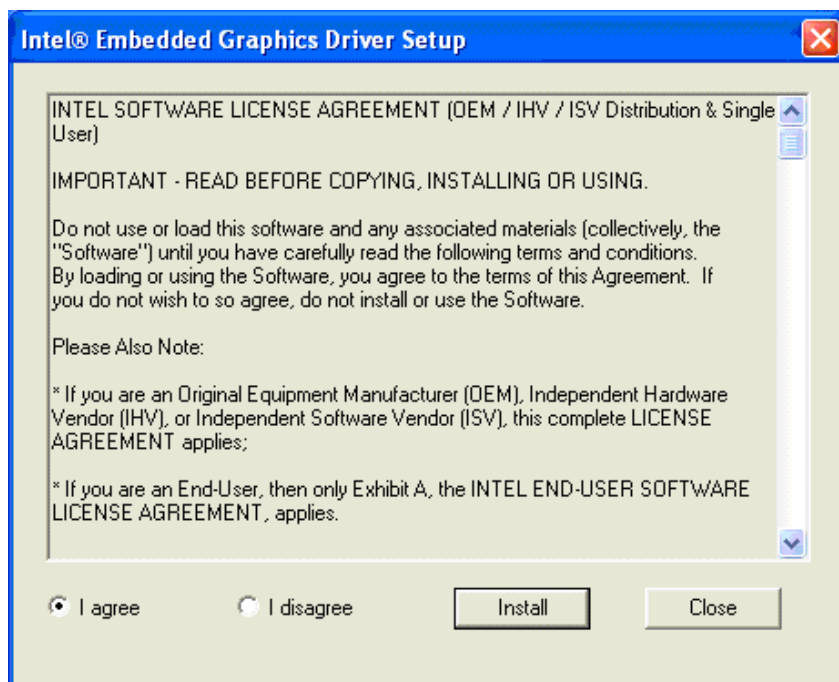
Warning: If you have a previous version of the IEGD installed on your system, you must remove it using the Microsoft Windows* **Add or Remove Programs** utility located in the **Control Panel**. Do not use the 5.0 version of the IEGD Install program to uninstall previous versions of the driver. If you do, unpredictable results may occur. You can use this program only to uninstall the driver from Version

5.0. Each version of the driver has its own version of the installer/uninstaller utility.

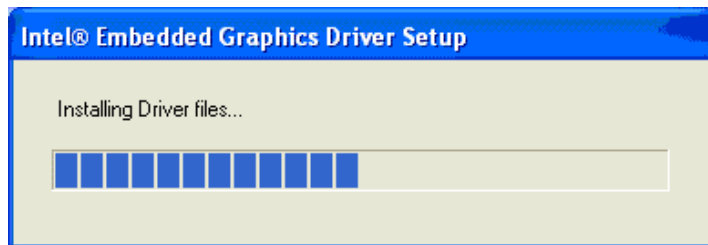
1. Double click the `setup.exe` icon in the `utilities` folder. The following dialog appears.



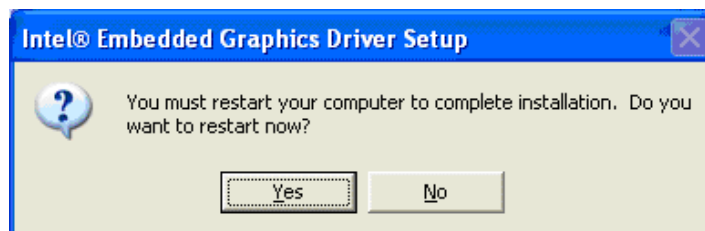
2. To install the driver, make sure that the **Installs driver and application files** radio button is selected, then click the **Next** button. The accept license screen appears.



3. Click the **I agree** radio button, then click the **Install** button. The installation begins and shows a progress bar as follows:



4. Once the driver and application files have been copied, the system must be restarted to complete the installation. If you want the installation program to restart your computer, click the **Yes** button.



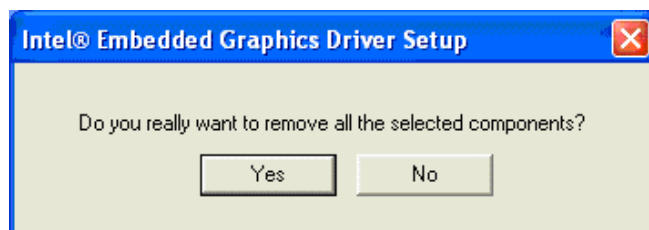
6.4 Uninstalling Version 5.0 of the Driver

You can use the `setup.exe` Microsoft Windows* GUI program to remove the driver from your system. When you run the uninstaller program, it removes the following items from the system:

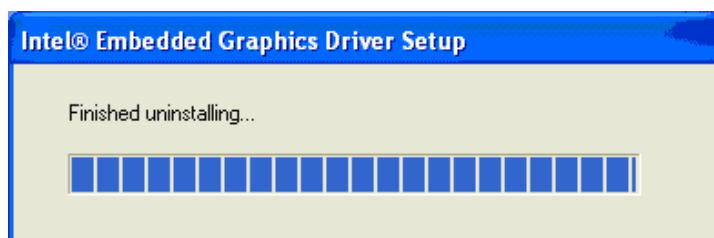
- The IEGD
- The `.inf` and `.pnf` files from the `windows\system32\inf` folder.
- The `DisplayPage.dll` and `qt-mt332.dll` from the `windows\system32` folder
- Data registry items by running `regsvr32.exe` with the `uninstall` option.

Warning: Never use the `setup.exe` program to remove a pre-5.0 version of the driver. If you do, unpredictable results may occur. Always use the **Add or Remove Programs** utility in the **Control Panel** to remove a previous version of the IEGD. You can only use this utility to remove version 5.0 of the driver. Each version of the driver has its own version of the installer/uninstaller utility.

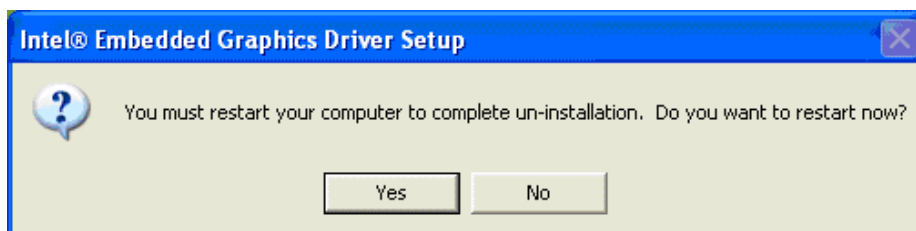
1. Click the `setup.exe` icon located in the `utilities` subfolder of the `IEGD_5_0_Windows` folder.
2. In the dialog box, select the **Uninstalls driver and application files** radio button, then click the **Next** button. The following prompt appears:



3. Click the **Yes** button to remove the driver. A progress bar displays and when the driver has been removed, the following screen appears.



4. To complete the uninstallation, you must restart your system. If you want to restart your system now, click the **Yes** button in the following dialog.



6.5 Run-Time Operation

Resolution, refresh rate, and color bit depth can be changed after installation and reboot via a Microsoft Windows* display property sheet. On Microsoft Windows* 2000 and Microsoft Windows XP*, extended desktop can be enabled and disabled, along with swapping primary and secondary displays. Other operations such as enabling and disabling ports (display output), rotation, port configuration, and attribute control are accessible via the standard display driver escape protocol.

6.6 Viewing and Changing the Driver Configuration From Microsoft Windows*

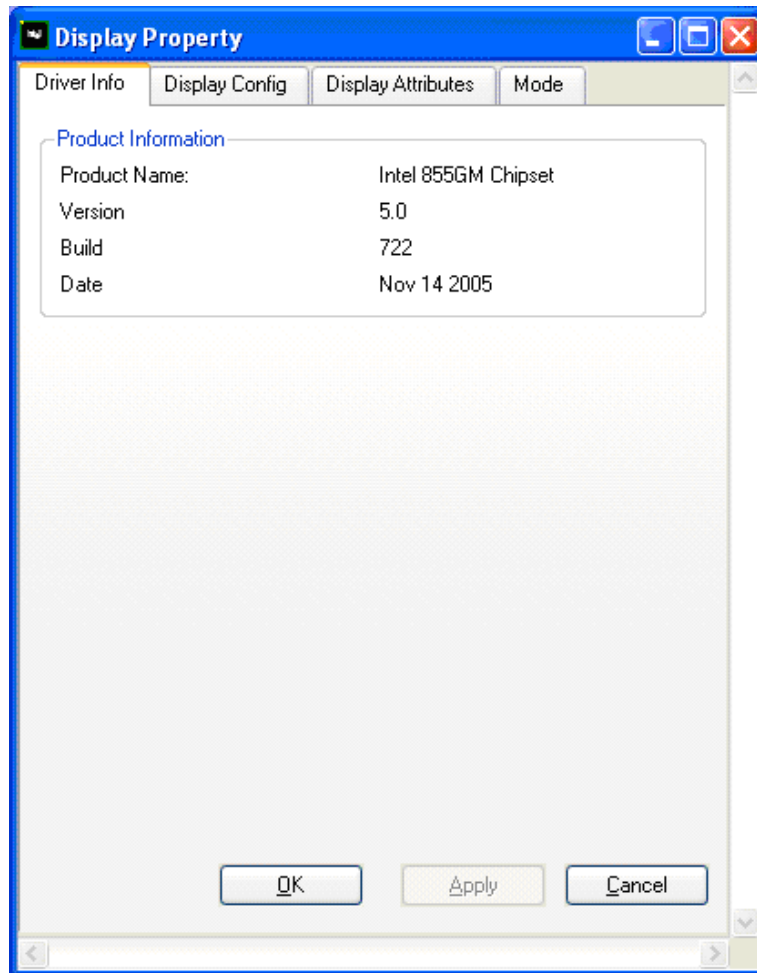
You can change certain configuration attributes of the IEGD using the `IEGDGUI.exe` program located in the `IEGD_5_0_Windows\utilities` folder. On Microsoft Windows XP* and Microsoft Windows* 2000 systems, you can access the IEGD Configuration tabs through the **Advanced Settings** tab of the **Display Properties** icon from the Windows **Control Panel**. This program launches the IEGD Configuration GUI that consists of the following four tabs:

- **Driver Info** -- Contains the driver status and display information.
- **Display Config** -- Contains current display information and allows configuration of display modes, flip, rotation, and enabling/disabling for a given port.
- **Display Attributes** -- Contains the supported Port Driver (PD) attributes and allows configuration of PD attributes.
- **Mode** -- Contains current resolution and bit depth settings for the primary and secondary ports. You can change the settings for both the primary and secondary ports from this tab.

To view or change the driver settings using the GUI interface, follow this procedure.

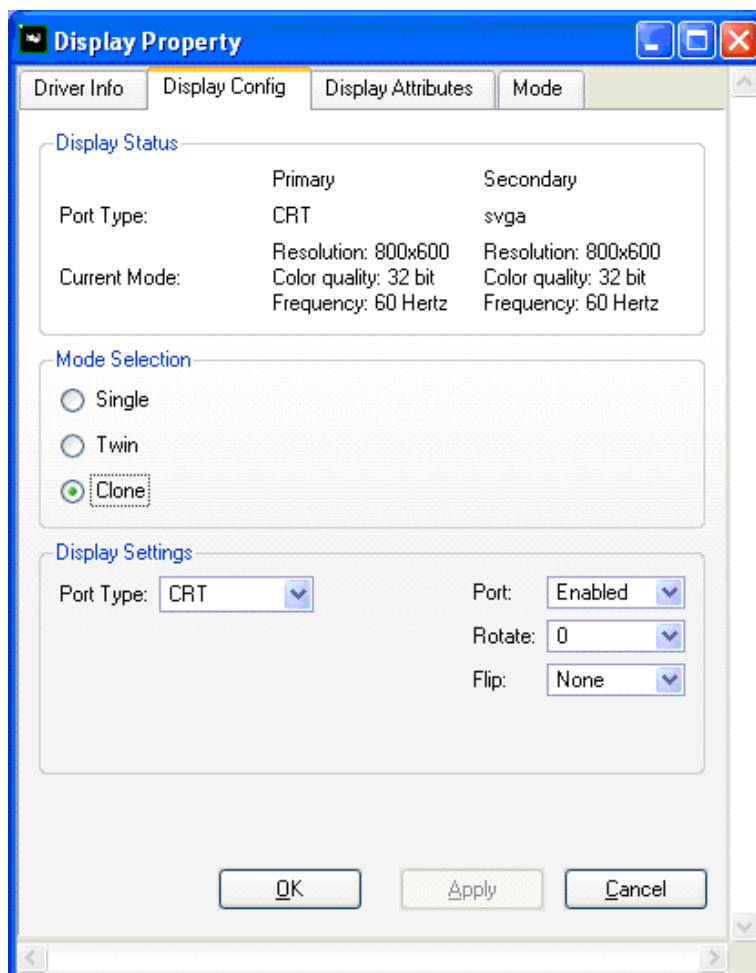
1. Double click the IEGDGUI .exe icon in the utilities folder. On Microsoft Windows XP* and 2000 systems, you can click the **Advanced Settings** tab in the **Display Properties** icon from the **Control Panel**. The IEGD Configuration GUI appears and shows information about the driver.

Figure 17. Runtime Configuration GUI -- Driver Info Tab



2. Click the **Display Config** tab to show the current configuration.

Figure 18. Runtime Configuration GUI -- Display Config Tab



The **Display Status** section of the dialog shows the current configuration for the **Primary** and **Secondary** displays.

3. You can change the Display Mode of the display between **Single**, **Twin**, and **Clone** by clicking the appropriate radio button in the **Mode Selection** section of the dialog.
4. To view and change the settings for a display, select the **Port Type** from the listbox in the **Display Settings** section of the dialog. You can then change the following Settings:
 - **Port**: Allows you to enable or disable the port.
 - **Rotate**: You can rotate the display 0, 90, 180, and 270 degrees.
 - **Flip**: Inverts the display horizontally.

If you change any configuration settings in the **Display Config** dialog box, click the **Apply** button for the changes to take effect.

5. Click the **Display Attributes** tab to view and change the attributes for the display. The screen that appears depends upon your current configuration. The following screenshots show several different possible screens that appear when the **Display Attributes** tab is clicked.

Figure 19. Runtime Configuration GUI -- Display Attributes Tab

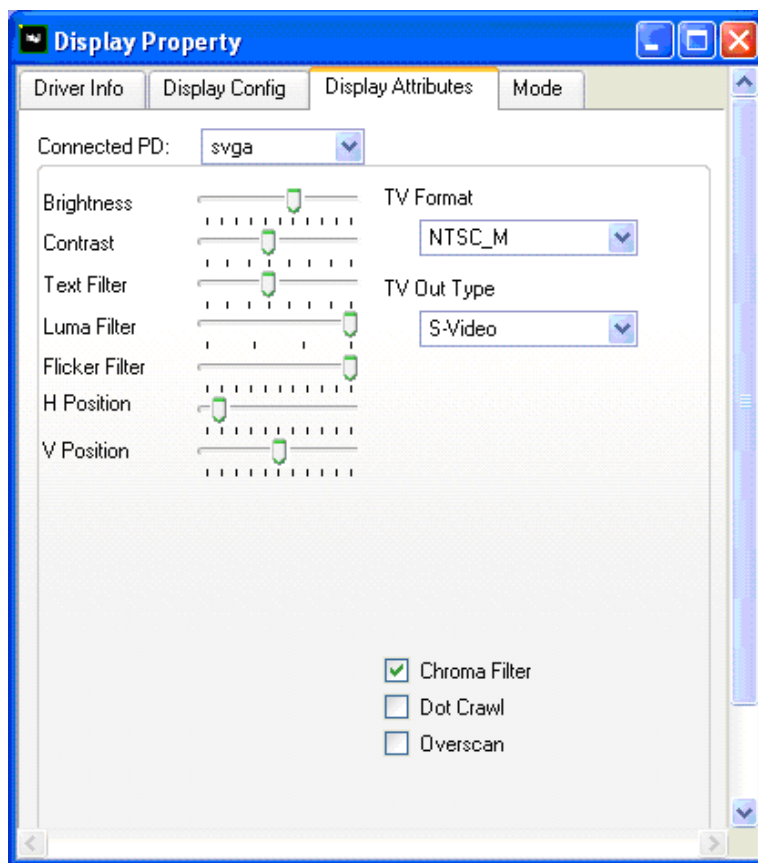
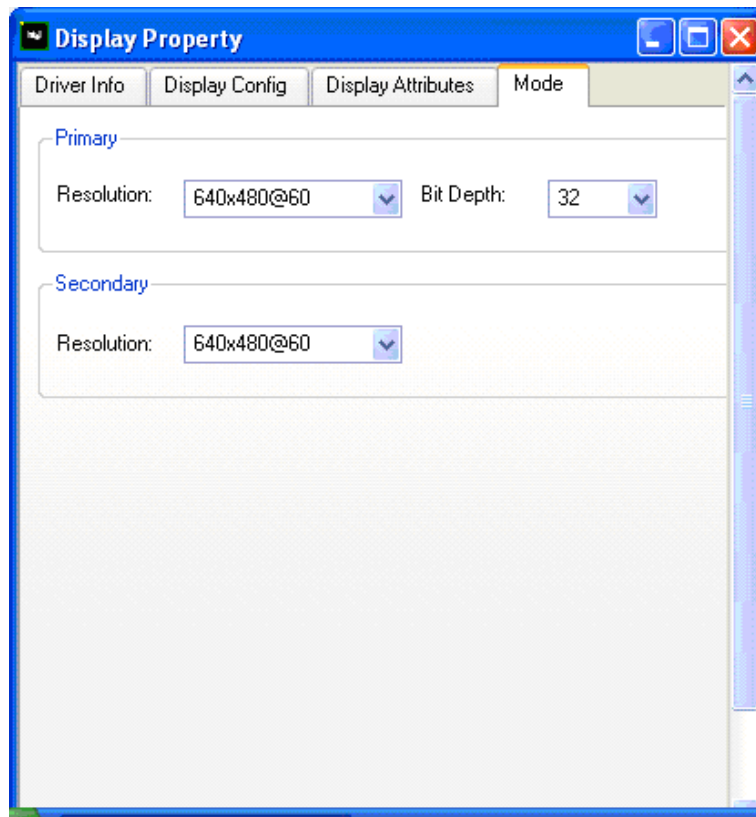


Figure 19 shows the attributes that can be changed for a TV connected to a CH7009 Digital Transmitter. The **Connected PD** shows the current Port Driver selected for the display. You can change the Port Driver by selecting the appropriate one for your device. The attributes that appear on this tab depend upon the selected port driver. Refer to [Appendix C, “Port Driver Attributes,”](#) for a complete list of port driver attributes.

6. Click the **Mode** tab to view and change the current mode’s resolution and bit depths. The screen that appears depends upon your current configuration. [Figure 20](#) shows a sample Mode tab screen that has both a primary and secondary display.

Figure 20. Runtime Configuration GUI -- Mode Tab Screen



7. If you make any changes to the mode settings, click the **Apply** button to have the changes take effect.

This page intentionally left blank.

Installing and Configuring Microsoft Windows CE* Drivers

7

7.1 Overview

This chapter describes the driver-level information for Microsoft Windows CE* .Net operating systems.

7.2 Microsoft Windows CE* .NET Installation

The following sections describe how to install the IEGD on the Microsoft Windows CE* .NET 4.2 and 5.0 operating systems.

7.2.1 Prerequisites

The development system should have the following software installed:

- Microsoft Windows XP* Professional, SP2
- Platform Builder for Microsoft Windows CE* .Net 4.2 or 5.0 (with latest service packs)

The target system must contain one of the following Intel chipsets:

- Intel[®] 945G Chipset
- Intel[®] 945GM Chipset
- Intel[®] 915GV Chipset
- Intel[®] 915GM Chipset
- Intel[®] 855GME Chipset
- Intel[®] 852GM Chipset
- Intel[®] 852GME Chipset
- Intel[®] 845GV Chipset
- Intel[®] 815 Chipset
- Intel[®] 815E Chipset

The target system must also contain a minimum of 64MB of RAM.

7.2.2 Integrating IEGD With Microsoft Windows CE* .NET Platform Builder

The integration/installation of the driver binaries depends upon the requirements of the target device; while `ddi_igd.dll` is required, port drivers may be optionally included.

To integrate the driver binaries into the Platform Builder, the catalogue file included with the release should first be imported to the Platform Builder's catalogue. After that, find "Intel Embedded Graphics Driver" under Third Party -> Device Drivers -> Display and Drag-And-Drop it into the BSP.

In order for the Platform Builder to use IEGD, the `video.reg` file included with the release has to be properly included into the BSP. For WinCE 4.2, this means appending the content of `video.reg` into `project.reg`. For WinCE 5.0, this means adding the following lines into the `platform.reg`. Note that you must specify the correct path to the `video.reg` file.

```
;*****
;* IEGD Section for WinCE 5.0 Platform Builder
;******
IF BSP_DISPLAY_IEGD

; @CESYSGEN IF CE_MODULES_PCI

; GWES will be able to auto-detect the display adapter if a candidate value
; points to the adapter's PCI instance. Values from Candidate1 to Candidate32
; (decimal) are possible; GWES will examine them sequentially until it finds
; a match.
[HKEY_LOCAL_MACHINE\System\GDI\DisplayCandidates]
    "Candidate6"="Drivers\\Display\\Intel"

[($PCI_BUS_ROOT)\Template\IEGD]
"DisplayDll"="ddi_igd.dll"
"Class"=dword:03
"SubClass"=dword:00
"ProgIF"=dword:00

"VendorID"=multi_sz:"8086", "8086", "8086", "8086", "8086", "8086", "8086", "8086",
"8086", "8086", "8086", "8086", "8086"
"DeviceID"=multi_sz:"3582", "2572", "2562", "357B", "3577", "1132", "7125", "7123",
"7121", "2582", "2782", "2592", "2792"

; include the path to the video.reg file in the release package
#include <specify_path_here>\video.reg

; @CESYSGEN ENDIF CE_MODULES_PCI
ENDIF ; BSP_DISPLAY_IEGD
```

Finally, to include the actual driver binaries into the OS image, you must reference them in the BSP's BIB file. For WinCE 4.2, this means going into **Platform Settings -> BIB Info** and adding `ddi_igd.dll`, `igd3dalm.dll` along with any other port drivers you want to include. For WinCE 5.2, this means appending the path to `ddi_igd.dll` and the port drivers into `platform.bib`, as shown below. Note that `igd3dalm.dll` is not required for WinCE 5.0 because IEGD currently does not support D3D Mobile.

Figure 21. Sample FILES Block from platform.bib File

FILES	Name	Path	Memory Type
;	-----	-----	-----
;			
;	@CESYSGEN IF CE_MODULES_DEVICE		
;	@CESYSGEN ENDIF CE_MODULES_DEVICE		
	ddi_igd.dll	\$(_FLATRELEASEDIR)\ddi_igd.dll	NK
	ch7009.dll	<specify_path_here>\ch7009.dll	NK
	ch7017.dll	<specify_path_here>\ch7017.dll	NK
	fs454.dll	<specify_path_here>\fs454.dll	NK

7.3 Microsoft Windows CE* .NET Configuration

The following sections describe how to configure the IEGD on the Microsoft Windows CE* .NET 4.2 and 5.0 operating systems.

7.3.1 Mode Setting Through the Registry

The `video.reg` file contains registry entries for the IEGD and must be present for the driver to work. Display configurations are set through the registry. The driver reads these entries at startup and attempts to validate it with a list of known modes. If the validation is successful, the drivers sets the mode; otherwise, the configuration defaults to a Width and Height of 640x480, a Depth of 16 bits per pixel (bpp), and a refresh rate of 60Hz.

The following registry keys are Microsoft Windows CE* .NET specific and are configured in the `project.reg` file. All non-ASCII keys must be entered in hexadecimal notation. Unless otherwise specified, all keys reside under the following registry path:

[HKEY_LOCAL_MACHINE\Drivers\Display\Intel]

All keys follow this syntax:

"<keyname>"=dword:<value>

The following example specifies a Display Mode of 640 x 480 pixels, 32-bit color depth, and a refresh rate of 60Hz.

```
[HKEY_LOCAL_MACHINE\DRIVERS\Display\Intel]
"Width"=dword:280
"Height"=dword:1E0
"Depth"=dword:20
"Refresh"=dword:3C
```

The driver reads these entries at startup and attempts to validate it with a list of known valid modes. If the validation is successful, it sets the mode; otherwise, it defaults to 640x480x16x60.

The following options allow for configuring the video memory location and size to a static location. Modifying these requires strong understanding of the `config.bib` file. The example `config.bib` file that comes with the release works with the values being set for these options.

If either of the below values is zero or they are not defined, then the dynamic memory allocation method is used for video memory usage. Thus availability is dependent on system resources. This is in compliance with the Dynamic Video Memory Technology.

```
[HKEY_LOCAL_MACHINE\Drivers\Display\Intel]
;"ReservedMemoryBase"=dword:3200000
    ; starting physical address of video memory
;"ReservedMemorySize"=dword:0E00000
    ; size of video memory (includes stolen memory)
```

Note: The above values and current defaults in the video.reg file are configured to use the last 14 of 64MB of RAM on the system. Please refer to [Section 7.3.4, “Determining Graphics Memory Size”](#) on page 109 for details on this static video memory configuration.

For more information on advanced modes and configurations that are available, refer to [Section 7.3.8, “Display Mode Configuration”](#) on page 111.

Table 20 lists all the registry keys for a Microsoft Windows CE* configuration.

Table 20. Registry Keys (Sheet 1 of 4)

Registry Entry	Description	Possible Ranges
Width Height	Width and Height of the display.	Width and Height must be expressed as hexadecimal values. For example: 1024 x 768: 400 x 300 800 x 600: 320h x 258 640 x 480: 280 x 1E0
Depth	Color depth in bpp (bits per pixel)	Depth must be expressed as a hexadecimal number and must be one of the following values: 8bpp: 8 16bpp: 10 24bpp: 18 32bpp: 20 (Note that the 815 chipset does not support 32bpp and the 845, 855, and 915 chipsets do not support 24 bpp.)
Refresh	The refresh rate of the display.	Refresh rate must be in hex: 60 : 3c 70 : 46 75 : 4b 85 : 55 etc... This value can be any valid refresh rate as long as the display port supports it. A refresh of '0' takes the first refresh that matches width, height and depth.

Table 20. Registry Keys (Sheet 2 of 4)

Registry Entry	Description	Possible Ranges
PortOrder	<p>PortOrder must be specified as a quoted string containing four digits. The valid values are:</p> <ul style="list-style-type: none"> 1 - Integrated TV Encoder (915GM only) 2 - DVO/sDVO B port 3 - DVO/sDVO C port (or DVO port on 815/E) 4 - Integrated LVDS port (852GME, 855GME, or 915GM only) 5 - Analog CRT port <p>Default: 0 for all keys</p>	<p>Search order for detecting attached displays for the Display Detection feature. When Display Detection is enabled, the PortOrder determines which display is primary and which display is secondary.</p> <p>The port search order can be specified to ensure the port device (DVO device) is found, based on the system integrator's routing choices. Default ordering is chosen by specifying zeros in the PortOrder keys.</p> <p>Default ordering is chipset specific; see Table 32, "Default DVO Search Order" on page 180.</p> <p>Refer to Section 3.7, "Display Detection and Initialization" on page 47 for more information on using PortOrder in combination with the Display Detect feature.</p>
edid	If VBIOS/Driver reads EDID from panel/CRT.	<p>0 – Do not read EDID from panel/CRT</p> <p>1 – Attempt to extract EDID timing data from panel/CRT</p>
edid_avail edid_not_avail	<p>These two parameters are used to control the available timings for any display. edid_avail is used when EDID values are read from the display. If an attempt to read EDID from the display fails or the edid parameter is set to 0, then the driver uses the edid_not_avail flags.</p> <p>The value for both parameters must be specified as a hex value.</p> <p>Defaults: edid_avail: 3 (hex). Bit 0 = 1, Bit 1 = 1, Bit 2 = 0 (Use driver built-in standard timings and EDID block and filter modes.)</p>	<p>edid_not_avail: 1 (hex). Bit 0 = 1, Bit 1 = 0, Bit 2 = 0. (Use driver-built-in standard timings.)</p> <p>Range [16 bits]</p> <p>Valid values (specified in hex):</p> <p>bit 0 ----- 0 - Do not use driver built-in standard timings 1 - Use driver built-in standard timings</p> <p>bit 1 (not applicable to edid_not_avail) ----- 0 - Do not use EDID block 1 - Use EDID block to filter modes</p> <p>bit 2 ----- 0 - Do not use user-defined DTDs 1 - Use user-defined DTDs</p> <p>bit3 - bit15 ----- Reserved for future use.</p>
NO_D3D	<p>Specify whether to enable d3d.</p> <p>Note: For WinCE 5.0, this must be set to 0 as IEGD currently does not support D3D on WinCE 5.0 platforms.</p>	<p>0 = Enable D3D 1 = Disable D3D</p> <p>Default is 0.</p>

Table 20. Registry Keys (Sheet 3 of 4)

Registry Entry	Description	Possible Ranges
DisplayConfig	The display configuration (single, twin, clone, or vertical extended display)	<p>DisplayConfig must be set to one of the following values:</p> <p>1 = Single 2 = Clone 4 = Twin 5 = VEXT</p> <p>(Note that the 815/815E and 845 chipsets do not support Clone and VEXT modes because they contain only one pipe.)</p>
DisplayDetect	Determines whether display detection is on or off	<p>DisplayDetect must be one of the following values:</p> <p>0 = Off (Default) 1 = On.</p> <p>(The DisplayDetect feature is used to detect a child display before using it. CAUTION: Setting this value to 1 means the display is not enabled if the detection fails.)</p>
ReservedMemoryBase ReservedMemorySize	Video memory can be statically reserved or dynamically allocated on demand. If both <i>ReservedMemoryBase</i> and <i>ReservedMemorySize</i> are non-zero, then Video memory allocation utilizes the static model.	<p>The ReservedMemoryBase plus the ReservedMemorySize must extend to the TOM (Top Of Memory) and not conflict with other reserved memory arenas in config.bib. Default for both base and size is zero, indicating a dynamic allocation model.</p> <p>Default behavior disables static memory model.</p>
MaxFbSize	Maximum size of the expected frame buffer. By providing this hint, the display driver can more efficiently organize GART memory, leading to a smaller video memory consumption.	<p>Must be greater than or equal to the expected size of frame buffer. Units are in bytes. Specifying zero causes the default frame buffer reservation sizing.</p> <p>Default: 815 Chipset: 12 Mbytes All other chipsets: 16 Mbytes</p>
MinVidSurfX MinVidSurfY	In pixels, the minimum width and height of surfaces in order to be acceptable for allocation in Video memory. Due to hardware restrictions that optimize memory access, it is advisable to reserve video memory for larger surfaces and allow GDI and DirectDraw* to allocate small surfaces from system memory.	<p>No limitations. Suggested values for both width and height are 10. Default value for both width and height is 1.</p> <p>Default: MinVidSurfX = 1 MinVidSurfY = 1</p>
i2cpin	The GPIO pin pair used as I2C bus to read and write to DVO device registers. For pin pair definition, see Table 10, "Pin Pair Definitions" on page 41 .	<0-6>
ddcpin	The GPIO pin pair used as DDC bus to read panel EDID data. For pin pair definition, see Table 10, "Pin Pair Definitions" on page 41 .	<0-6>

Table 20. Registry Keys (Sheet 4 of 4)

Registry Entry	Description	Possible Ranges
i2cdab	I2C Device Address Byte (dab) for reading and writing device registers. The device address byte should be in 8-bit format with the 7-bit slave address assigned to its bits 7:1 and bit 0 set to 0.	<0x00-0xff>
ddcdab	I2C device address for reading EDID data from panel through DDC bus.	<0x00-0xff>
i2cspeed	Speed of I2C bus for DVO device.	[10-400]. Units in KHz
ddcspeed	Speed of I2C bus for EDID device.	[10-400]. Units in KHz
SysToVidStretch	Enables system-to-video memory stretch blit operations to take advantage of hardware-accelerated filtering. Normally, it is more efficient to allow GDI to conduct system-to-video stretch blits, but the default filtering used by GDI is Nearest. See the <i>BlendFilter</i> key below for hardware accelerated filtering options.	0 = Disabled 1 = Enabled Default: 0
BlendFilter	Provides selection of hardware-accelerated filtering methods for stretch blit operations.	0 = Nearest 1 = Bilinear 2 = Anisotropic Default: 0
NoTear	If enabled, all blit operations to the frame buffer are synchronized with video sync to eliminate any visible tearing on the display screen. Disabling this feature achieves a performance gain.	0 = Disabled, tearing allowed 1 = Enabled, no visible tearing Default: 0
PortDrivers	List of DVO transmitter port drivers to load when the system boots.	Space separated string enclosed in quotes, where each port driver name is listed in the string. The default string included with the release has all supported port drivers: "ch7009 fs454 lvds ns2501 ns387 sii164 th164 sdvo"

7.3.2 Port Driver-Specific Installation Options

Port driver attributes can also be specified in the `video.reg` file and used to initialize the Port Driver during system boot. The keys are port driver-specific and reside under a different registry path than the general installation options:

```
[HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\2\Attr]
```

The display driver enumerates all keys under this path and passes them to the port driver. If the port driver does not support a specific key, then it is ignored.

Appendix C, “Port Driver Attributes” provides the attributes supported by each port driver.

7.3.3 Microsoft Windows CE* .NET Memory Model -- Reserved Memory

The Intel Embedded Graphics Suite (IEGS = VBIOS + Graphics driver) provides the ability to dedicate additional memory for graphics functions on the Microsoft Windows CE* .NET platform. This is known as *reserved memory*. The amount of reserved memory is selected by firmware. The reservation size is passed to the graphics driver through a scratch register available on the GMCH. Reserved memory is useful in minimizing the amount of memory stolen from the OS for memory-limited, embedded systems. For instance, if firmware utilizes a 640 x 480, 32-bit frame buffer, a total of 1.2 Mbytes is required. On an Intel® 845GV platform, stolen memory would need to be configured as 8 Mbytes, since the next smaller option is only 1 Mbyte, too small for the 640 x 480, 32-bit frame buffer. In such a case, stolen memory can be programmed to 1 Mbyte. The additional memory required for the frame buffer can then be provided by reserved memory, allowing a minimum amount of memory to be removed from the OS.

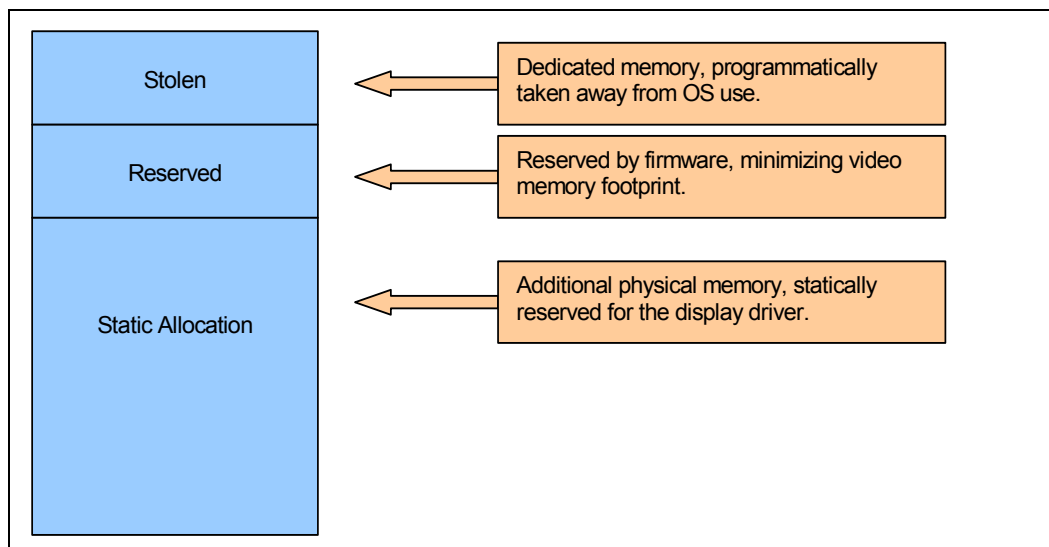
Note: Reserved memory is only available on the Microsoft Windows CE* .NET operating system, and must be accounted for in the `config.bib` memory layout file.

Additionally, the Microsoft Windows CE* .NET display driver can be configured for either static or dynamic allocation of video memory. The static model preallocates physical memory for the display driver and provides a more efficient surface allocation scheme. The dynamic model allocates surface memory on demand from the system and will incur a small performance hit. However, the dynamic model has the advantage of deallocation of video memory when not required, thus making it available to other applications.

The static memory model requires a base and size specification registered in the `project.reg` file. The base + size must reach to top of memory (TOM). Since this is not required to be specified in the `config.bib` memory map, care must be taken not to overlap any other memory arenas with the static allocation. See [Section 7.2, “Microsoft Windows CE* .NET Installation” on page 101](#) for further details on how to configure the static memory model.

Figure 22 shows a typical memory map, using a static memory model.

Figure 22. Typical Memory Map Using Static Memory Model



7.3.4 Determining Graphics Memory Size

The IEGD supports static as well as dynamic location of graphics memory. For static video memory, the two registry settings `ReservedMemoryBase` and `ReservedMemorySize` denote where and how big this area will be. This portion of memory will be taken care off by the built in gart driver. The size should include the stolen memory (BIOS setting, if applicable).

If either of these registry settings are zero or not defined in the registry, then dynamic memory allocation is used for video memory management. The usage will be dependent on system resources. This is in line with the Dynamic Video Memory Technology.

As an example of static video memory, if the stolen memory is 8 MBytes and you want a total of extra 56 MBytes of graphics memory for a total of 64 Mbytes of graphics memory you would want to have these settings: (On a 128 MByte machine)

```
[HKEY_LOCAL_MACHINE\Drivers\Display\Intel]
    "ReservedMemoryBase"=dword:4000000
    "ReservedMemorySize"=dword:4000000
```

Which means that the managed graphics memory pool will begin at physical address 0x4000000 (64 MBytes) and is 64 MBytes of size.

Note: Always remember to include the amount of stolen memory in this number.

The lower 64 MBytes in the above machine will be used for the NK.bin and RAM, so therefore you must change your `config.bib` accordingly. The configuration for the given machine is (debug build):

NK	80220000	01650000	RAMIMAGE
RAM	81700000	02790000	RAM

As you see, the NK.BIN image takes 22.3125 MBytes and the rest of 41.6875 MBytes is RAM. The release build should have much smaller NK area. For better control, users may turn `AUTOSIZE=OFF` to calculate exactly where the image and RAM starts and ends followed by the location of the video memory pool with no possibility of runtime deviation.

These two settings, together with the registry settings mentioned above will decide what your machines memory layout will look like.

The example `config.bib` and the memory configuration settings in the `video.reg` are validated settings that can be used for static video memory of 16 MBytes in size starting from the 48 MBytes address, thus including the stolen memory.

7.3.5 Framebuffer and Video Surface Size

Two additional optional registry settings are available to limit the `FrameBuffer` size of the display driver and the total size of offscreen video surfaces.

The `MaxFbSize` registry entry will control the maximum size of the frame buffer only. Actual usage will depend on the mode being used.

The `PageReqLimit` registry entry will control the total size in pages (4 KBytes) of all video surfaces, buffers allocated for any use. Both of these registry configurations apply to both the static as well as dynamic video memory management explained in the previous section. The default below indicates that a maximum of 2 MBytes are used for the frame buffer and a maximum of 16 MBytes are permitted for all offscreen videosurface allocations.

```
"MaxFbSize"=dword:200000
"PageReqLimit"=dword:1000
```

In the case of Microsoft Windows CE*, because the O.S. does not allow for dynamically setting the frame buffer size, the `MaxFbSize` can be changed to match the mode setting being used in order to minimize on video memory wastage. The following are different suggested values for `MaxFbSize` for different display modes. These values have not been validated. Note that 640x480 is calculated as 640x512 and 800x600 is calculated as 800x768 for stride alignment purposes.

```
640x512X16 = A0000
640x512X24 = F0000
640x512X32 = 140000
800x768X16 = 12C000
800x768X24 = 1C2000
800x768X32 = 258000
1024x768X16 = 180000
1024x768X24 = 240000
1024x768X32 = 300000
```

7.3.6 Video Surface Allocation Rule

Another two optional registries entries determine a minimum width and height dimension that allows video surface allocations to succeed.

In the Microsoft Windows CE* GDI, video surface allocations can happen with a `REQUIRE_VIDEO_MEMORY` or a `PREFER_VIDEO_MEMORY` flag. The following options force surface allocations with the `PREFER_VIDEO_MEMORY` flag to be allocated in system memory if the width and height are lower then what is stated in the following entries.

```
"MinVidSurfX"=dword:10
"MinVidSurfY"=dword:10
```

In this example, surfaces allocated with the `PREFER_VIDEO_MEMORY` where the width and height are both less than 16 pixels are forced to be in system memory.

7.3.7 Driver General Configuration Selection

The Intel® Embedded Graphics Drivers allow you to provide multiple instances of per port platform customizations. There can be up to 16 instances of configurations and a single variable, called `ConfigId`, ensures the display driver selects the right instance. Each instance may contain multiple groups of per-config and per-config+per-port platform customizations.

```
"ConfigId"=dword:1
; 1 = default
```

7.3.8 Display Mode Configuration

The "DisplayConfig" registry setting (see below), allows you to decide whether you want to configure the display configuration to be in single, twin, clone, or vertical extended modes. (Microsoft Windows CE* .Net does not support Extended mode like Microsoft Windows XP*). DisplayConfig is a per configuration setting, so note the registry path that the Config number must match what's chosen in ConfigId above.

```
[HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\General]
    "DisplayConfig"=dword:1
```

Note: The Intel® 815/E and Intel® 845GV chipsets do not support clone mode.

DisplayConfig does not, however, dictate what type of display ports will be used. The PortOrder setting ensure the correct display port types are used based on user selection. Again, these are per-configuration settings.

```
[HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\General]
;-----
; Select Port Order
;-----
"PortOrder"="5432"

; PortOrder specifies the actual of port
; that will be taken for the Primary /
; Secondary ports if there are duplicates
; of the same type. For example, if both
; Primary and Secondary are digital, then
; port order will which DVO ports will be
; first and second. The section below gives
; the port order numbers for various chipsets.
; Specify value "0000" to use default settings.
; On i915 chipsets:
; =====
; 1 - Integrated TV Encoder
; 2 - DVO B port/RGBA port
; 3 - DVO C port
; 4 - Internal LVDS port
; 5 - Analog port
;
; On i830/835/845/85x/865 chipsets:
; =====
; 1 - DVO A port
; 2 - DVO B port/RGBA port
; 3 - DVO C port
; 4 - Internal LVDS port
; 5 - Analog port
;
; On 835: If RGBA is used (DVO B & C together), then use DVO B number
; to specify any parameter for it.
;
; On i81x chipsets:
; =====
; Port numbers:
; 1 - DVO port
; 2 - Analog port
```

To ensure that the above display device configurations are NOT used when the actual child display device (panel/monitor) is NOT connected, use the "DisplayDetect" key to ensure that if the panel / monitor is not detected other ports are tried for the primary display at the least (ensuring that at least 1 truly connected display is available for view).

There is also an option to detect the displays before the actual initialization of the display ports to happen. However, this option must be used carefully. Displays without EDID will NOT be detected.

```
"DisplayDetect"=dword:0
; Set DisplayDetect to '1' to detect display ports
```

7.3.8.1 Vertical Extended Display Mode

The Windows CE* IEGD driver supports Vertical Extended Display Mode, which allows you to configure two displays by doubling the height of resolution. One large frame buffer is allocated. The top half of the frame buffer is on the first pipe and the bottom half is on the second pipe. The Windows CE* operating system is unaware of the two displays. This feature is supported only on the following dual-pipelined chipsets:

- Intel® 852GM Chipset
- Intel® 855GME Chipset
- Intel® 915GM Chipset

This feature is enabled through the `DisplayConfig` key in the `project.reg` file. The resolution, bit depth, and refresh rates of both displays must be the same. Vertical panning is *not* supported; horizontal panning is not. DirectDraw is supported on both pipes, but DirectDraw 3D must be disabled when Vertical Extended Display Mode is enabled.

The following example specifies Vertical Extended Display Mode enabled with the primary display set to an analog CRT in the `PortOrder`, the secondary display set to a DVI display, and both displays set to a resolution of 800 x 1200 pixels (the height is double from the normal 800 x 600 display mode), 32-bit color depth, and a refresh rate of 60Hz.

```
[HKEY_LOCAL_MACHINE\DRIVERS\Display\Intel]
"Width"=dword: 320
"Height"=dword:4B0
"Depth"=dword:20
"Refresh"=dword:3c
...

"NO_D3D"=dword:0
...

"DisplayConfig"=dword:5
...

"PortOrder"="5432"
```

7.3.9 Per Port Platform Customization

The Intel Embedded Graphics Drivers provide what is considered the most useful tools to the embedded market -- per port platform customizations. This includes the following:

- Defining custom DTD panel timings

```
: PixelClock, HorzActive, HorzSync etc...
```


- Customized GPIO pin selection for I2C and DDC communication with DVO encoders and panels.

```
:I2cPin, I2cDab, I2cSpeed etc...
```

- Flat Panel width and height limitations and power and/or backlight control mechanisms

```
:BkltMethod, BkltT1, BkltT2, GpioPinVdd etc...
```

- Port driver specific attribute settings for initialization at boot time.

```
:Brightness, Contrast, H-Position etc...
```

All of the above can be set for each individual port depending on the maximum number of ports the chipset supports. Also, you can have multiple instances of these configurations to allow different settings per configuration.

The usage model for this per-config, per-port platform customizations follows after the same options available in the INF registry settings for the Intel Embedded Graphics Drivers for Microsoft Windows XP*. Refer to [Figure 23, “Sample video.reg File” on page 118](#) or to the provided registry sample file in the IEGD Windows CE* .Net driver package for examples. The following sections provide information on these configurations.

7.3.9.1 Per Port Customization -- Custom DTD Mode Timings

For each configuration, each port can be added with up to 255 customized DTD modes. The following is an example of adding 800x640 mode to the LVDS port when Config=1 is used.

However, the DTD timings should be preceded with the following keys:

- Edid - (1=Read EDID from port, 0=Don't Read EDID)

```
[HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\2\General]
    "Edid"=dword:0

[HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\4\DTD\1]
    "PixelClock"=dword:9c40
    "HorzActive"=dword:320
    "HorzSync"=dword:28
    "HorzSyncPulse"=dword:80
    "HorzBorder"=dword:0
    "HorzBlank"=dword:100
    "HorzSize"=dword:0
    "VertActive"=dword:280
    "VertSync"=dword:1
    "VertSyncPulse"=dword:4
    "VertBorder"=dword:0
    "VertBlank"=dword:1c
    "VertSize"=dword:0
    "Flags"=dword:1e
```

Note: For Per-Config, Per-Port configuration, the subkey path includes the correct Config and Port numbers

7.3.9.2 Per Port Customization -- Custom DVO GPIO Pin Settings

For each configuration, each port's GPIO pin pair settings can be configured in terms of which physical pins, what I2C slave address the DVO encoder on that port responds to and what speeds to use.

```
; [HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\2\DVO]
;   "I2cPin"=dword:2
;   "I2cDab"=dword:70
;   "I2cSpeed"=dword:0
;   "DdcPin"=dword:0
;   "DdcSpeed"=dword:0
```

Note: For Per-Config, Per-Port configuration, the subkey path includes the correct Config and Port numbers.

7.3.9.3 Per Port Customization -- Custom Flat Panel Controls

Similarly, the flat panel native resolution and power and backlight sequencing controls can also be configured here.

```
; [HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\4\FPInfo]
;   "Width"=dword:320
;   "Height"=dword:280
;   "BkltMethod"=dword:0
;   "BkltT1"=dword:0
;   "BkltT2"=dword:0
;   "BkltT3"=dword:0
;   "BkltT4"=dword:0
;   "BkltT5"=dword:0
;   "GpioPinVdd"=dword:0
;   "GpioPinVee"=dword:0
;   "GpioPinBklt"=dword:0
;   "BkltEnable"=dword:0
;   "UseGMCHClockPin"=dword:0
;   "UseGMCHDataPin"=dword:0
```

Note: For Per-Config, Per-Port configuration, the subkey path includes the correct "Config" and "Port" numbers

7.3.9.4 Per Port Customization -- Attribute Initialization

Attributes are also per config and per port. However, the actual keys are dependant on the port driver being used. Below are examples of registry keys associated with initializing attributes for the Chronitel* Port Driver.

Attribute Ranges for the CH7009 Port Driver:

Brightness	0-100
Contrast	0-7
Flicker Filter	0-4
Saturation	0-7
Hue	0-100

Text Filter	0-3	
Macrovision	boolean	
Overscan ratio	1-4	(Low, Std, High, None)
TV Format	1, 17, 33	(NTSC-M, NTSC-M-J, NTSC-4.33)
TV Output	1-4	(Comp & Svid, Comp, Svid, Comp)

For complete information on port driver attributes, refer to [Appendix C](#).

Note: For Per-Config, Per-Port configuration, the subkey path includes the correct "Config" and "Port" numbers.

The following example sets the CH7009 port driver attributes using the attribute IDs. Refer to [Section C.2.1, "Chrontel* CH7009/CH7010 Port Driver TV Attributes" on page 165](#) for a list of CH7009 attribute IDs and their meanings.

```
[HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\2\Attr]
    "0"=dword:32
    "1"=dword:4
    "3"=dword:1
    "8"=dword:1
    "12"=dword:0
    "14"=dword:1
    "19"=dword:1
```

7.3.10 System to Video Stretch Blit

System to Video Memory stretch blits are not supported on Intel GMCH devices. This feature allows you to enable a soft copy of system surfaces to video surfaces in order to conduct an accelerated stretch blit. The advantage of this is that the stretch blit then utilizes the blend engine and hardware filtering can be applied. The filtering options are listed below under "Blend Filtering".

A value of 1 for the "SysToVidStretch" enables system to video stretch blits, as described above, while a value of 0, disables this feature and forwards all system to video stretch blits to the emulator provided by the operating system.

```
[HKEY_LOCAL_MACHINE\Drivers\Display\Intel]
    "SysToVidStretch"=dword:0
```

7.3.11 Blend Filtering

The blend filtering method can be selected via the BlendFilter registry setting. The filter method chosen will be used for all stretch blit operations, including video to video stretch blits, as well as system to video stretch blits. The default filtering method is Nearest.

Available Blend Filter options:

```
0 == Default == Nearest
1 == Bilinear
2 == Anisotropic
3 == 4x4
```

```
[HKEY_LOCAL_MACHINE\Drivers\Display\Intel]
"BlendFilter"=dword:2
; Blend filter to use for all stretch blits
; BlendFilter 0 == NEAREST
; BlendFilter 1 == BILINEAR
; BlendFilter 2 == ANISOTROPIC
; BlendFilter 3 == 4X4
```

7.3.12 Catalogue Feature File

A Catalogue Feature File, `igd.cec`, is available. In order to add this driver into the Platform Builders catalogue feature list, take the following steps:

1. Create the following folder WINCE420 PUBLIC root folder:

```
C:\WINCE420\PUBLIC\COMMON\OAK\DRIVERS\DISPLAY\INTELGID
```

2. Copy the `ddi_igd.dll`, `ddi_igd.rel` and `video.reg` files into that folder
3. Install the `igd.cec` file into the Platform Builder by:
 - a. From the File menu, select **Manage Catalog Features**.
 - b. - Choose **Import**.
 - c. - In the **Import Catalog Features** dialog box, select the `.cec` file, and then click **Open**.
 - d. - From the **View** menu, select **Catalog** to display the Catalog.

7.3.13 Text Anti-Aliasing

The Microsoft Windows CE* .Net driver supports text anti-aliasing. To switch it on, add these registry settings:

```
[HKEY_LOCAL_MACHINE\System\GDI\Fontsmoothing]
[HKEY_LOCAL_MACHINE\System\GDI]
    "ForceGRAY16"=dword:1
```

Note: Text Anti-Aliasing should always be turned on when using a TV display device.

7.3.14 Dynamic Loading of Port Drivers

You can load port drivers dynamically through the `PortDrivers` registry entry. The `PortDrivers` registry setting determines which port drivers to be dynamically loaded. The port drivers are loaded in the order specified in the registry entry. These dll's must exist in the `c:\Windows` directory.

7.3.15 NOTEARING Support for Smooth Blits

The IEGD provides NOTEARING support to give flicker free images. The `TearFB` registry setting enables or disables the tearing option.

The default setting is OFF "0" (i.e., tearing is enabled). Enabling `TearFB` results in a performance penalty.

7.3.16 video.reg File Backward Compatibility

Version 5.0 of the IEGD provides backward compatibility with pre-5.0 versions of the `video.reg` file. This support is implemented through the new `PcfVersion` key in the `video.reg` file, shown below:

```
[HKEY_LOCAL_MACHINE\Drivers\Display\Intel]
"PcfVersion"=dword:400
```

The IEGD uses this key to determine which version of the `.inf` file it is interpreting. When this key is present in the `video.reg` file and its value is 400, the driver reads it as a 5.0 `video.reg` file. If this key is omitted from the `video.reg` file or if its value is less than 400, the driver reads the `video.reg` file as a pre-5.0 file.

Note the following rules:

- If you use a pre-5.0 version of the `video.reg` file with version 5.0 of the IEGD, the driver translates pre-5.0 configuration parameters to 5.0 parameters.
- You cannot use 5.0 parameters in a pre-5.0 `video.reg` file. If you try, the driver ignores them.
- You cannot use pre-5.0 parameters in a 5.0 `video.reg` file. If you try, the driver ignores them.

For example, the `usestdtimings` parameter is a pre-5.0 parameter. If it is specified in a 5.0 `video.reg` file, the driver ignores it. Similarly, if you attempt to add the `NO_D3D` parameter to a pre-5.0 `video.reg` file (that is, an `video.reg` file where the `PcfVersion` key is not present), they are ignored by the driver.

The default `video.reg` file for version 5.0 already contains the `PcfVersion` key. There is usually no need to add it to the `.inf` file.

Figure 23. Sample video.reg File

```

;***** BEGIN INTEL DISPLAY DRIVER REGISTRY ENTRY *****
;*****

[HKEY_LOCAL_MACHINE\System\GDI\Drivers]
"Display"="ddi_igd.dll"

[HKEY_LOCAL_MACHINE\Drivers\Display\Intel]
;-----
; Following registry entry for
; version used
; 400 : IEGD 5.0 version
"PcfVersion"=dword:400
;-----
; Following registry entries for
; display settings: resolution,
; bit depth and refresh rate
;-----
"Width"=dword:320
"Height"=dword:258
; Width & Height values must be hex
; 1280x1024 : 500h x 400h
; 1024x768 : 400h x 300h
; 800x600 : 320h x 258h
; 640x480 : 280h x 1E0h
; etc...
; VEXT MODE height is doubled
; 640x960 : 280h x 3c0
; 800x600 : 320h x 4b0h
; etc...; can be other standard timings
; as long as the display device
; and chipset supports it

"Depth"=dword:10

; Bit depth must be one of:
; 8bpp : 8
; 16bpp : 10
; 24bpp : 18
; 32bpp : 20
; (815 does not support 32 bpp)
; (83x,845,855,865,915 does not support 24 bpp)

"Refresh"=dword:3c
; Refresh rate must be in hex:
; 60 : 3c
; 70 : 46
; 75 : 4b
; 85 : 55
; etc...
; any refresh rate as long as
; the display port supports it
; refresh of '0' will take the
; first refresh that matches
; width, height and bpp

;-----
; Following is registry entry for
; controlled configuration of
; video memory usage / location
;-----

; "ReservedMemoryBase"=dword:03200000
; "ReservedMemorySize"=dword:00E00000
; Above settings are for a 64M platform,
; where the video memory is 14M at the top
; the above settings are assuming there is
; no system bios / firmware that has stolen
; memory from top of memory. If it does exist
; reduce ReservedMemorySize avoiding overlap
; exception for ACSFL, memory area is reused
; NOTE: CURRENTLY THESE SETTINGS ARE REMARKED
; FOR DYNAMIC VIDEO MEMORY CONFIGURATION

```

```

"MaxFbSize"=dword:300000
; Above is Maximum Frame Buffer Size
; used to limit the maximum size in bytes
; of the main frame buffer

"PageReqLimit"=dword:8000
; Page Request Limit is used to control
; the max allocations of offscreen video
; surfaces, buffers etc..
; value is in number of pages (4K).
; this is independent of dynamic or static
; memory configuration.
; The max for 830m = 128MB = 0x8000
; The max for 815 = 64MB = 0x4000

"MinVidSurfX"=dword:10
"MinVidSurfY"=dword:10
; Above settings are to define a minimum
; width and height that would allow for video
; surface allocations to succeed.
; eg: surfaces with width < 16 are
;     forced to be in system-mem
; eg: surfaces with height < 16 are
;     forced to be in system-mem
; only affects allocations of surfaces
; with GPE_PREFER_VIDEO_MEMORY flag

"ConfigId"=dword:1
; This value dictates the configuration
; to select for Per-Port settings from port
; specific registry. The settings mirror
; Windows XP IEGD drivers implementation

; *****
; Following are the registry
; entries for acceleration
; configuration
; *****
; -----
; Set SysToVidStretch to '1' to enable
; driver to perform System to Video stretch
; blits (for 815, this is not supported)
; -----
"SysToVidStretch"=dword:1
; -----
; Blend filtering method selection
; The hardware must be capable of
; support, else, emulation is done.
; Possible blend methods are:
;   BlendFilter 0 == NEAREST
;   BlendFilter 1 == BILINEAR
;   BlendFilter 2 == ANISOTROPIC
;   BlendFilter 3 == 4X4
; (for 815, this value does not
; have any effect since currently
; the driver does not support
; any form of accelerated stretch
; blits on the 815)
; -----
"BlendFilter"=dword:2
; -----
; Provide a list of port drivers to attempt to load upon boot time
; -----
"PortDrivers"="ch7009 ch7017 fs454 lvds ns2501 ns387 sii164 th164 sdvo"
; -----
; Option for enabling/disabling
; TEARING - Default is OFF
; -----
; Set '1' to enable the NOTEARING option
"TearFB"=dword:1
; -----
; Specify whether to enable d3d
; NO_D3D Value: 0 (default)
;               : 0 --> Enable D3D
;               : 1 --> Disable D3D

```

```

;-----
"NO_D3D"=dword:0
;*****
; The Following Sections Provide Per-Config
; & Per-Port Registry Settings
;*****
;-----
; Following are the registry
; entries for the port device
; configuration - per-Config
;-----
[HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\General]
;-----
; Select Display configuration, single, twin ...
;-----
"DisplayConfig"=dword:1
; Possible Display Config combo:
;   DisplayConfig 1 == SINGLE
;   (Single is default if none specified)
;   DisplayConfig 4 == TWIN
;   --> (Twin mode: common timing across ports)
;   DisplayConfig 2 == CLONE
;   --> (Clone mode: distinct timing per port)
; (815,845 does not support clone mode - only 1 pipe)
;   DisplayConfig 5 == VEXT (vertical extend)
;   --> (Vert Extended modes : "Height" )
;   ( registry key value must be 2X the )
;   ( intended port timings. Both ports )
;   ( must use the same timings. For )
;   ( example, for port timings of )
;   ( 800x600, the DisplayConfig should )
;   ( be 5 and the Height=1200 or 0x4b0 )
;   ( Overlay wont work in VEXT mode. )
;-----
; Select if u want to enable Display Detection
;-----
"DisplayDetect"=dword:0
; DisplayDetect is to detect display child device
; before using it. BEWARE, setting this to '1' will
; mean display wont be enabled if detection failed

;-----
; Select Port Order
;-----
"PortOrder"="5432"

; PortOrder specifies the actual of port
; that will be taken for the Primary /
; Secondary ports if there are duplicates
; of the same type. For example, if both
; Primary and Secondary are digital, then
; port order will which DVO ports will be
; first and second. The section below gives
; the port order numbers for various chipsets.
; Specify value "0000" to use default settings.
; On i915 chipsets:
; =====
;   1 - Integrated TV Encoder
;   2 - DVO B port/RGBA port
;   3 - DVO C port
;   4 - Internal LVDS port
;   5 - Analog port
;
; On i830/835/845/85x/865 chipsets:
; =====
;   1 - DVO A port
;   2 - DVO B port/RGBA port
;   3 - DVO C port
;   4 - Internal LVDS port
;   5 - Analog port
;
; On 835: If RGBA is used (DVO B & C together), then use DVO B number
; to specify any parameter for it.
;

```



```
; On i81x chipsets:
; =====
; Port numbers:
; 1 - DVO port
; 2 - Analog port

; The following sections provide the various registry
; settings for Per-Port custom DTD timings, per port DVO
; I2C or panel power control as well as Attribute setups
; If these are not provided, defaults are used,
; - Use standard VESA timings
; - Use ADD-Card specifications for I2C
; - No panel power or backlight control
; - No Attribute settings at init time.

;*****
; The sections below are for the more detailed per port
; registry configurations. It follows the same usage model and
; key value meanings as the Windows INF registry configuration
; file. Refer to IEGD User Manual Sections 5.4.1 to 5.4.4
; for details of this.
;*****
;-----
; Config 1 - DVO-B Port (For Almador) |
;-----
; Following are the registry
; entries for port's general config
;-----
;
; EdidAvail and EdidNotAvail: <only 16 bits used>
; -----
; These 2 parameters can be used to control the available timings for
; any display. 'EdidAvail' is used when EDID is read from the display
; device. If an attempt to read EDID is failed or 'Edid = 0' then
; driver uses 'EdidNotAvail' flags.
;
; See below bit definitions for both 'EdidAvail' and 'EdidNotAvail'
;
; BIT 0:
; -----
; 0 - Do not use driver built-in standard timings
; 1 - Use driver built-in standard timings
;
; BIT1: <not applicable to EdidNotAvail>
; -----
; 0 - Do not use EDID block
; 1 - Use EDID block and filter modes
;
; BIT2:
; -----
; 0 - Do not use user-DTDs
; 1 - Use user-DTDs.
;
; BIT3-BIT15
; -----
; Future use.
;
; Default behavior:
; -----
; If user doesn't provide EdidAvail and EdidNotAvail, then
; EdidAvail = Use Std timings + Use EDID block and Filter modes
; EdidNotAvail = Use Std timings
; [HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\2\General]
; "Edid"=dword:1
; "EdidAvail"=dword:7 ; STD TIMINGS + EDID TIMINGS + USER TIMINGS
; "EdidNotAvail"=dword:7 ; STD TIMINGS + USER TIMINGS
;-----
; Following are the registry entries
; for port's DVO I2C settings
;-----
; [HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\2\DVO]
; "I2cPin"=dword:2
; "I2cDab"=dword:70
; "I2cSpeed"=dword:0
; "DdcPin"=dword:0
```

```

; "DdcSpeed"=dword:0
; -----
; Following are the registry entries
; for port's flat panel's mode-limits,
; power and backlight control
; -----
; [HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\2\FPInfo]
; "Width"=dword:0
; "Height"=dword:0
; "BkltMethod"=dword:3
; "BkltT1"=dword:1E
; "BkltT2"=dword:4
; "BkltT3"=dword:4
; "BkltT4"=dword:14
; "BkltT5"=dword:1F4
; "GpioPinVdd"=dword:27
; "GpioPinVee"=dword:26
; "GpioPinBklt"=dword:28
; "BkltEnable"=dword:1
; "UseGMCHClockPin"=dword:0
; "UseGMCHDataPin"=dword:0
; -----
; Following are the registry entries
; for ports first custom DTD mode to add
; -----
; [HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\2\DTD\1]
; "PixelClock"=dword:9c40
; "HorzActive"=dword:320
; "HorzSync"=dword:28
; "HorzSyncPulse"=dword:80
; "HorzBorder"=dword:0
; "HorzBlank"=dword:100
; "HorzSize"=dword:0
; "VertActive"=dword:280
; "VertSync"=dword:1
; "VertSyncPulse"=dword:4
; "VertBorder"=dword:0
; "VertBlank"=dword:1c
; "VertSize"=dword:0
; "Flags"=dword:1e
; -----
; Following are the registry entries
; for ports second custom DTD mode to add
; (Up to 255 can be added)
; -----
; [HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\2\DTD\2]
; "PixelClock"=dword:9c40
; "HorzActive"=dword:320
; "HorzSync"=dword:28
; "HorzSyncPulse"=dword:80
; "HorzBorder"=dword:0
; "HorzBlank"=dword:100
; "HorzSize"=dword:0
; "VertActive"=dword:258
; "VertSync"=dword:1
; "VertSyncPulse"=dword:4
; "VertBorder"=dword:0
; "VertBlank"=dword:1c
; "VertSize"=dword:0
; "Flags"=dword:1e
; -----
; Following are the registry
; entries for the port device'
; display attribute parameters
; Use when enabling Port device
; example below is for Conexant
; on Port2 (DVO-B for almador)
; key names depend on port driver
; -----
; [HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\2\Attr]
; id 0=dword:32
; id 1=dword:4
; id 3=dword:1
; id 15=dword:4
; id 2=dword:32

```

```
; id 12=dword:0
; id 13=dword:0
; id 19=dword:1
; id 8=dword:1
; id 14=dword:1
; "Composite and S-Video"=dword:1

;-----
; Config 1 - Analog Port (For Almador) |
;-----
[HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\5\General]
; "Edid"=dword:1
; "EdidAvail"=dword:7 ; STD TIMINGS + EDID TIMINGS + USER TIMINGS
; "EdidNotAvail"=dword:7 ; STD TIMINGS + USER TIMINGS
[HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\5\DTD\1]
; "PixelClock"=dword:9c40
; "HorzActive"=dword:320
; "HorzSync"=dword:28
; "HorzSyncPulse"=dword:80
; "HorzBorder"=dword:0
; "HorzBlank"=dword:100
; "HorzSize"=dword:0
; "VertActive"=dword:280
; "VertSync"=dword:1
; "VertSyncPulse"=dword:4
; "VertBorder"=dword:0
; "VertBlank"=dword:1c
; "VertSize"=dword:0
; "Flags"=dword:1e
[HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\5\DTD\2]
; "PixelClock"=dword:9c40
; "HorzActive"=dword:320
; "HorzSync"=dword:28
; "HorzSyncPulse"=dword:80
; "HorzBorder"=dword:0
; "HorzBlank"=dword:100
; "HorzSize"=dword:0
; "VertActive"=dword:258
; "VertSync"=dword:1
; "VertSyncPulse"=dword:4
; "VertBorder"=dword:0
; "VertBlank"=dword:1c
; "VertSize"=dword:0
; "Flags"=dword:1e

;-----
; Config 1 - Int-LVDS Port (For Almador) |
;-----
[HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\4\General]
; "Edid"=dword:1
; "EdidAvail"=dword:7 ; STD TIMINGS + EDID TIMINGS + USER TIMINGS
; "EdidNotAvail"=dword:7 ; STD TIMINGS + USER TIMINGS
[HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\4\FPIInfo]
; "Width"=dword:320
; "Height"=dword:280
; "BkltMethod"=dword:0
; "BkltT1"=dword:0
; "BkltT2"=dword:0
; "BkltT3"=dword:0
; "BkltT4"=dword:0
; "BkltT5"=dword:0
; "GpioPinVdd"=dword:0
; "GpioPinVee"=dword:0
; "GpioPinBklt"=dword:0
; "BkltEnable"=dword:0
; "UseGMCHClockPin"=dword:0
; "UseGMCHDataPin"=dword:0
[HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\4\DTD\1]
; "PixelClock"=dword:9c40
; "HorzActive"=dword:320
; "HorzSync"=dword:28
; "HorzSyncPulse"=dword:80
; "HorzBorder"=dword:0
```

```

; "HorzBlank"=dword:100
; "HorzSize"=dword:0
; "VertActive"=dword:280
; "VertSync"=dword:1
; "VertSyncPulse"=dword:4
; "VertBorder"=dword:0
; "VertBlank"=dword:1c
; "VertSize"=dword:0
; "Flags"=dword:1e
; [HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\4\DTD\2]
; "PixelClock"=dword:9c40
; "HorzActive"=dword:320
; "HorzSync"=dword:28
; "HorzSyncPulse"=dword:80
; "HorzBorder"=dword:0
; "HorzBlank"=dword:100
; "HorzSize"=dword:0
; "VertActive"=dword:258
; "VertSync"=dword:1
; "VertSyncPulse"=dword:4
; "VertBorder"=dword:0
; "VertBlank"=dword:1c
; "VertSize"=dword:0
; "Flags"=dword:1e

; [HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1]
; "name"="IEGD SDVO Configuration File"
; [HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\2]
; "name"="svga"
; [HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\2\FPIInfo]
; For a SDVO driver, sample settings for the panel : 1400x1050
; "Width"=dword:578
; "Height"=dword:41A
; [HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Config\1\Port\2\Attr]
; "27"=dword:1

; enable font anti-aliasing
[HKEY_LOCAL_MACHINE\System\GDI\Fontsmoothing]

[HKEY_LOCAL_MACHINE\System\GDI]
"ForceGRAY16"=dword:1
;***** INTEL DISPLAY DRIVER REGISTRY ENTRY END
*****;*****

```

Linux Installation and Configuration 8

This chapter describes the configuration and installation of the IEGD for Linux systems. The IEGD supports X-servers from the X.org* and XFree86* organizations.

8.1 Overview

The IEGD Linux distribution package contains drivers built for the following X-Servers:

- XFree86 version 4.2.0
- XFree86 version 4.3.0
- X.org version 6.7.0
- X.org version 6.8.2

The IEGD requires the official version or these servers from the www.Xfree86.org or the www.X.org websites and may not operate with other versions of these servers.

8.2 Prerequisites

The following list shows the items that you must have before you can configure and install the Linux driver.

- Platform with supported Intel chipset.
- Platform with a minimum of 128 MBytes.
- Display device specifications.
- Driver package consisting of the following directories and files that are located under the IEGD_5_0_Linux directory. (Note that in the pathnames that follow, <xserver name> is the directory name of the X-Server version of the driver. For example, XFree86-4.2, XFree86-4.3, or X.org-6.7.0.)
 - Documents/ReInotes
 - Documents/Errata
 - Documents/License
 - Documents/<xserver name>/intel.4
 - Documents/<xserver name>/IntelDriverCtl.3x
 - Documents/<xserver name>/IntelPortCtl.3x
 - Driver/<xserver name>/intel_drv.o
 - Driver/<xserver name>/libXportctl.a
 - Driver/<xserver name>/libXportctl.so.1.0
 - Driver/<xserver name>/portctl.h
 - Driver/<xserver name>/libXiegdctl.a
 - Driver/<xserver name>/libXiegdctl.so.1.0
 - Driver/<xserver name>/iegdctl.h

```
Driver/<xserver name>libXiegd_escape.a
Driver/<xserver name>libXiegd_escape.so.1.0
```

```
- Utilities/gui_lib_setup_linux.txt
Utilities/IEGDGUI
Utilities/pcf2iegd
```

- The following files are located in the `IEGD_Patches` directory:
 - `agpgart.patch-2.4.20` (Patch for 2.4.20 kernel)
 - `agpgart.patch-2.4.20-8` (Patch for Redhat* 2.4.20-8 kernel)
 - `agpgart.patch-2.4.24` (Patch for 2.4.24 kernel)
 - `agpgart.patch-2.6.5-1.358` (Patch for Fedora* 2.6.5 kernel)
 - `COPYING` (GPL* license agreement for patches)
- Linux kernel source tree for active running kernel
- Linux kernel installed with module support enabled.
- Linux kernel installed with `agpgart` module enabled.
- XFree86 version 4.3, XFree86 version 4.3, X.org version 6.7.0, or X.org version 6.8.2 X-Server installed and functional.
 - Directory paths to XFree86 or X.org installation and configuration files.
- System administration privileges.

8.3 Installation

To install the IEGD, you must first obtain the `IEGDLinux.tgz` file from the following website:

http://www.intel.com/design/intarch/swsup/graphics_drivers.htm

To get the Linux IEGD, click the Intel® Embedded Graphics Driver for Linux - v5.0 link and follow the instructions for downloading the `.tgz` file.

Once you have the `IEGDLinux.tgz` file, you can install the IEGD by performing the following steps:

1. Log into to a system administration account.
2. Untar the driver package to a convenient location.

```
tar -xvzf <driver package.tgz>
```

This creates a directory structure in the directory where you extracted the `.tgz` file and contains the following directories:

- `IEGD_5_0_Linux` -- Contains the `Documents`, `Driver`, `License`, and `Utilities` subdirectories.

The `Documents` subdirectory contains the `Xfree86` and `Xorg-X11` subdirectory. These directories contain man pages for the IEGD.

The `Driver` directory contains subdirectories for `XFree86-4.2`, `XFree86-4.3`,

X.org-6.7, and X.org-6.8.

The `Utilities` directory contains IEGD utilities, including the `pcf2iegd` utility and the `IEGDGUI` runtime configuration utility.

- `IEGD_Patches` -- Contains files for patching the Linux kernel for agpgart.

3. Check the version of the X-Server your system is running. If you are running the XFree86 server, type the following command:

XFree86 -version

If you are running Xorg server, type the following command:

Xorg -version

4. Copy the IEGD driver binary, `intel_drv.o`, from the `IEGD_5_0_Linux/Driver/<xserver name>` directory to the X-Server's `modules/drivers` directory. The default installation directory is `/usr/X11R6/lib/modules/drivers`. This location can vary by distribution so check your system for the proper path. For example, if you are installing to an XFree86 version 4.3 X-Server, type the following commands:

```
cd ../IEGD_5_0_Linux/Driver/XFree86-4.3/  
cp intel_drv.o /usr/X11R6/lib/modules/drivers
```

5. Copy the necessary port driver files (`*.so` files in the `../IEGD_5_0_Linux/Driver/<xserver name>` directory) to the X-Server `lib/modules` directory. The default installation location is `/usr/X11R6/lib/modules`. This location can vary, so check your system for the proper path. Once the required port drivers have been copied, you can specify them in the `PortDrivers` option in the `Device` section of the config file. For more information on specifying the `PortDrivers` option, refer to [Table 21, “Supported Driver Options” on page 136](#). For example, to copy all the port drivers, type the following command:

```
cp *.so /usr/X11R6/lib/modules
```

6. Copy the port control client libraries (`libXportctl.a` and `libXportctl.so.1.0`) and the driver control client libraries (`libXiegdctl.a` and `libXiegdctl.so.1.0`) and the escape control libraries (`libXiegd_escape.a` and `libXiegd_escape.so.1.0`) from the `IEGD_5_0_Linux/Driver/<xserver name>` directory to the X-Server library directory. The default installation location is `/usr/X11R6/lib`. For example,

```
cp libXportctl.a libXportctl.so.1.0 libXiegdctl.a libXiegdctl.so.1.0 libXiegd_escape.a  
libXiegd_escape.so.1.0 /usr/X11R6/lib
```

7. In the X-Server library directory, create symbolic links for the port control and driver control library aliases:

```
cd /usr/X11R6/lib  
ln -s libXportctl.so.1.0 libXportctl.so  
ln -s libXportctl.so.1.0 libXportctl.so.1  
ln -s libXiegdctl.so.1.0 libXiegdctl.so  
ln -s libXiegdctl.so.1.0 libXiegdctl.so.1  
ln -s libXiegd_escape.so.1.0 libXiegd_escape.so  
ln -s libXiegd_escape.so.1.0 libXiegd_escape.so.1
```

8. Copy the port control and driver control include files (`portctl.h` and `iegdctl.h`) to the XFree86 `include/extensions` directory. The default installation location is `/usr/X11R6/include/X11`.

```
cd ../IEGD_5_0_Linux/Driver/XFree86-4.3
cp portctl.h iegdctl.h /usr/X11R6/include/X11
```

9. Copy the port control and driver control man pages (IntelPortCtl.3x and IntelDriverCtl.3x) from the IEGD_5_0_Linux/Documents/<xserver name> to the X-Server man/man3 directory. The default installation location is /usr/X11R6/man/man3. For example,

```
cd ../IEGD_5_0_Linux/Documents/XFree86
cp IntelPortCtl.3x IntelDriverCtl.3x /usr/X11R6/man/man3
```

10. Create the following hard links in the X-Server man/man3 directory:

```
cd /usr/X11R6/man/man3
ln IntelPortCtl.3x IntelPortCtlQueryExtension.3x
ln IntelPortCtl.3x IntelPortCtlQueryVersion.3x
ln IntelPortCtl.3x IntelPortCtlPortControl.3x
ln IntelPortCtl.3x IntelPortCtlGetPortAttributes.3x
ln IntelPortCtl.3x IntelPortCtlSetPortAttributes.3x
ln IntelPortCtl.3x IntelPortCtlReadPortAttributes.3x
ln IntelPortCtl.3x IntelPortCtlWritePortAttributes.3x
ln IntelPortCtl.3x IntelPortCtlQueryPort.3x
ln IntelPortCtl.3x IntelPortCtlQueryPorts.3x
ln IntelPortCtl.3x IntelPortCtlGetPortMode.3x
ln IntelPortCtl.3x IntelPortCtlGetPortModes.3x
ln IntelPortCtl.3x IntelPortCtlSetPortMode.3x
ln IntelPortCtl.3x IntelPortCtlSetPortModes.3x
ln IntelPortCtl.3x IntelPortCtlAllocPort.3x
ln IntelDriverCtl.3x IntelDriverCtlQueryExtension.3x
ln IntelDriverCtl.3x IntelDriverCtlQueryVersion.3x
ln IntelDriverCtl.3x IntelDriverCtlPortControl.3x
ln IntelDriverCtl.3x IntelDriverCtlGetDriverInfo.3x
```

11. From the IEGD_5_0_Linux/Documents directory, copy the driver man page, intel.4, to the man/man4 directory. The default installation location is /usr/X11R6/man/man4. This location can vary by distribution so check your system for the proper path. For example:

```
cd ../IEGD_5_0_Linux/Documents/XFree86
cp intel.4 /usr/X11R6/man/man4
```

12. Modify your XF86Conf or Xorg.conf file to include a device section for this driver and a Monitor section for your display. See [Section 8.4](#) for details on the driver configuration and the list of supported options. The default installation location for this file is /etc/X11

13. Patch the agpgart module with the Intel extensions:

To patch the Linux 2.4.2x and SuSE kernel with the GART changes:

1. cd into the kernel source directory (e.g., **cd /usr/src/linux-2.x.xxx**)
2. Execute the patch command. For example:

```
patch -p1 < ../IEGD_Patches/Driver/agpgart.patch-2.x.xxx
```

To update the kernel:

1. cd to the kernel source directory (e.g., **cd /usr/src/linux-2.x.xxx**)
2. Execute the make modules command.

make modules

3. Install the modules.

make modules_install

To patch a Linux 2.6.xxxx kernel with the GART changes for Fedora Core 2 kernels:

1. cd into the kernel source directory (e.g. **cd /usr/src/linux-2.6.5-1.358**)
2. Execute the patch command as follows:

patch -p1 < ../IEGD_Patches/Driver/agpgart.patch-2.6.xxxx

To update the kernel:

1. cd to the kernel source directory (e.g. **cd /usr/src/linux-2.6.5-1.358**)
2. Execute the make modules command:

make modules

3. Install the modules:

make modules_install

4. Run the following commands for the linux-2.6.xxxx kernel:

modprobe agpgart
modprobe intel-agp

Note: Since the agpgart is built-in by default in Fedora Core 2, it needs to be configured (eg. via **make menuconfig**) to install as a module before updating the kernel.

Also, to ensure the modules are loaded after a reboot, add the following line to the `/etc/modprobe.conf` file:

alias char-major-10-175 intel-agp

14. Set the LD_LIBRARY_PATH environment variable as follows:

export LD_LIBRARY_PATH=/usr/X11R6/lib:/usr/lib/qt-3.1/lib

The `/usr/lib/qt-3.1/lib` path is required for running the Runtime Configuration GUI (see [Section 8.5.1, “Runtime Configuration GUI \(IEGDGUI\)”](#) on page 143).

8.4 Configuration

The Intel Linux driver is for use with the integrated graphics of Intel chipsets on the Embedded Intel Architecture roadmap. The driver supports 8-, 16- and 24-bit pixel depths, dual independent head setup (only with the Intel® 852GME, Intel® 855GME, Intel® 915GM, and Intel® 915GM chipsets), flat panel, hardware 2D acceleration, hardware cursor, the XV extension, and the Xinerama extension.

8.4.1 Supported Hardware

The Intel driver supports the following chipsets with integrated graphics.

- Intel® 815
- Intel® 815E
- Intel® 845GV
- Intel® 852GM
- Intel® 852GME
- Intel® 855GME
- Intel® 915GV
- Intel® 915GM
- Intel® 945G
- Intel® 945GM

8.4.2 Configuration Overview

The driver auto-detects all device information necessary to initialize the integrated graphics device in most configurations. However, you can customize the IEGD configuration for any supported display by editing the X-server's configuration file (`XF86Conf` or `Xorg.conf`). Please refer to the `XF86Config(5x)` or `Xorg(5x)` man page for general configuration details. This section only covers configuration details specific to the IEGD.

In order to configure the IEGD for Linux, you must edit the X server's configuration file (`XF86Conf` or `Xorg.conf`). You can either edit the configuration directly or can use the `pcf2iegd` utility to create configurations that must then be copied into the configuration file. If you use the `pcf2iegd` utility, you must still edit the configuration file.

The following sections describe each method of configuring the IEGD for Linux.

8.4.2.1 Linux Configuration Using the `pcf2iegd` Utility

You can configure the Linux driver settings using the `pcf2iegd` utility as described in [Chapter 3, "Platform Configuration"](#). You can use either the `-x` or `-bin` option of the `pcf2iegd` utility to produce output for the `XF86Conf` or `Xorg.conf` file. The `-x` option is the preferred method for Linux systems.

Example: `pcf2iegd -x 1 system.pcf > system.config`

In this example, `system.pcf` is the name of the PCF file and the output file is named `system.conf`. The output file contains the settings required to configure the IEGD for Linux systems and can be pasted into the appropriate sections of the `XF86Conf` or `Xorg.conf` file

Alternatively, you can use the `pcf2iegd` utility to convert the PCF settings to the EDID parameter block files and example settings for the `XF86Conf` or `Xorg.conf` file. You must then edit the Linux configuration file and incorporate the settings contained in the output file from the `pcf2iedg` utility.

Example: `pcf2iegd -bin 1 system.pcf > system.config`

The example above generates two files. The first is the `system.config` file, which are sample settings that may be used in the `XF86Conf` or the `Xorg.conf` file. Please follow the specific setting options and values specified in [Table 21, “Supported Driver Options” on page 136](#), as the current version of the `pcf2iegd` utility may output options and values that do not apply to the Linux configuration.

The second file that is produced from the above example is the EDID parameter block file `EDID-1.bin`. This is a binary file that contains the EDID parameter information that is described in the PCF file associated with the configuration block with the ID of 1. It also contains all the configuration details for any display port, such as `FPInfo`, `DTDs`, `Attributes`, and other per-port general settings.

Note: The configuration ID in the `-bin` option of the `pcf2iegd` utility is optional. If omitted, the first configuration block in the PCF file is used, regardless of the configuration ID of that block. If your PCF file contains multiple configuration blocks, the `pcf2iegd` utility must be run for each configuration ID, and all resulting `EDID-n.bin` files should be transferred to the `/etc/X11` directory. See [Section 8.4.8, “EDID-Less Bin Configuration File” on page 140](#) for more details on these EDID parameter block files.

8.4.2.2 Editing the Linux Configuration Directly

Alternatively, you can edit the `XF86Conf` or `Xorg.conf` file directly. The following procedure outlines the steps to follow when editing the Linux configuration file. [Section 8.4.3, “The Linux Configuration File” on page 134](#) provides details on each section of the configuration file.

1. Login as root and open the configuration file for editing. The configuration file is typically located in the `/etc/X11` directory but may be located elsewhere on your system.
2. In the `Device` section of the configuration file, enter the appropriate information for your driver. The configuration file must have at least one `Device` section. The `Device` section lets you define information about the Intel graphics driver. You can use a single `Device` section for single, twin, or clone configurations. For Dual Independent Head configurations, you must specify a second `Device` section.
3. In the `Screen` section, enter information for each display in your configuration. The configuration file must have at least one `Screen` section. The `Screen` section binds a `Device` with a `Monitor` and lets you define resolution modes for the display. The `Screen` section is referenced in the `ServerLayout` section of the configuration file.
4. In the `Monitor` section, define monitor specifications and timings that will be used for the display. You must have a `Monitor` section defined for each display in your configuration. The `Monitor` section is referenced by the `Screen` section.
5. Save your changes to the file and reboot the system.

Figure 24 shows a complete, sample XF86Conf file configured to run with the Intel driver in clone mode. For detailed information on the configuration file, refer to the XF86Conf or Xorg man page. Section 8.4.3, “The Linux Configuration File” on page 134 describes the XF86Conf file in detail.

Figure 24. Sample XF86Conf File

```
# XFree86 4 configuration created by redhat-config-xfree86

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

Section "Files"
# RgbPath is the location of the RGB database. Note, this is the name of the
# file minus the extension (like ".txt" or ".db"). There is normally
# no need to change the default.
# Multiple FontPath entries are allowed (they are concatenated together)
# By default, Red Hat 6.0 and later now use a font server independent of
# the X server to render fonts.
    RgbPath         "/usr/X11R6/lib/X11/rgb"
    FontPath        "unix/:7100"
EndSection

Section "Module"
    Load            "dbe"
    Load            "extmod"
    Load            "fbdevhw"
    Load            "glx"
    Load            "record"
    Load            "freetype"
    Load            "type1"
EndSection

Section "InputDevice"
# Specify which keyboard LEDs can be user-controlled (eg, with xset(1))
# Option "Xleds"        "1 2 3"
# To disable the XKEYBOARD extension, uncomment XkbDisable.
# Option "XkbDisable"
# To customise the XKB settings to suit your keyboard, modify the
# lines below (which are the defaults). For example, for a non-U.S.
# keyboard, you will probably want to use:
# Option "XkbModel" "pc102"
# If you have a US Microsoft Natural keyboard, you can use:
# Option "XkbModel" "microsoft"
#
# Then to change the language, change the Layout setting.
# For example, a german layout can be obtained with:
# Option "XkbLayout" "de"
# or:
# Option "XkbLayout" "de"
# Option "XkbVariant" "nodeadkeys"
#
# If you'd like to switch the positions of your capslock and
# control keys, use:
# Option "XkbOptions" "ctrl:swapcaps"

# Or if you just want both to be control, use:
# Option "XkbOptions" "ctrl:nocaps"
#
    Identifier      "Keyboard0"
    Driver           "keyboard"
    Option           "XkbRules" "xfree86"
    Option           "XkbModel" "pc105"
    Option           "XkbLayout" "us"
EndSection
```

```

Section "InputDevice"
    Identifier "Mouse0"
    Driver "mouse"
    Option "Protocol" "IMPS/2"
    Option "Device" "/dev/psaux"
    Option "ZAxisMapping" "4 5"
    Option "Emulate3Buttons" "no"
EndSection

Section "InputDevice"
# If the normal CorePointer mouse is not a USB mouse then
# this input device can be used in AlwaysCore mode to let you
# also use USB mice at the same time.
    Identifier "DevInputMice"
    Driver "mouse"
    Option "Protocol" "IMPS/2"
    Option "Device" "/dev/input/mice"
    Option "ZAxisMapping" "4 5"
    Option "Emulate3Buttons" "no"
EndSection

Section "Monitor"
    Identifier "Monitor0"
    VendorName "Monitor Vendor"
    ModelName "Eizo T550"
    DisplaySize 290220
    HorizSync 30.0 - 82.0
    VertRefresh 50.0 - 160.0
    Option "dpms"
EndSection

Section "Monitor"
    Identifier "Monitor1"
    VendorName "SamSung"
    ModelName "SyncMaster 192MP"
EndSection

Section "Device"
    Identifier "IntelIEGD-1"
    Driver "intel"
    VendorName "Videocard vendor"
    BoardName "Intel 855"
    VideoRam 16384
    Option "PortOrder" "5300"
    Option "PortDrivers" "ch7009"
    Option "CloneDisplay" "Enable"
    Option "CloneRefresh" "60"
    Option "CloneWidth" "1024"
    Option "CloneHeight" "768"
    BusID "0:2:0"
EndSection

Section "Screen"
    Identifier "Screen0"
    Device "IntelIEGD-1"
    Monitor "Monitor0"
    DefaultDepth 16
    SubSection "Display"
        Depth 16
        Modes "800x600" "640x480"
    EndSubSection
EndSection

Section "DRI"
    Group 0
    Mode 0666
EndSection

```

8.4.3 The Linux Configuration File

To configure the IEGD for use with Linux, you must edit the Linux configuration file (XF86Conf or Xorg.conf). There are several sections within the configuration that must be edited or created, including:

- Device Sections
- Screen Sections
- Monitor Sections
- ServerLayout Section (when configuring DIH)
- ServerFlags Section (when configuring Xinerama)

The following sections describes each of these sections for use with the IEGD. Refer to the XF86Conf or Xorg.conf man pages for complete details.

8.4.3.1 Device Section

The Device section provides a description of a graphics device. The Linux configuration file (XF86Conf or 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. If you are creating a Dual Independent Head (DIH) configuration, you must have at least two Device sections.

The Device sections in the XF86Conf and Xorg.conf configuration files have the following format:

```
Section "Device"
    Identifier "devname"
    Driver "intel"
    entries
    ...
EndSection
```

The Identifier field defines the device. This name is used to associate the device with a screen and is referenced in Screen sections.

The Driver field defines the driver to use and is a required field in the Device section. The intel driver, intel_drv.o, must be installed in the /usr/X11R6/lib/modules/drivers directory.

The remainder of the Device section can contain IEGD-specific options. Refer to [Table 21 on page 136](#) for a list and description of IEGD supported options.

8.4.3.2 Screen Section

The Screen section is used to bind a Screen with a Device and a Monitor. It is also used to define resolution modes, color depths, and various other screen characteristics. Refer to the XF86Conf or Xorg man page for detailed information.

The Screen section has the following format:

```

Section "Screen"
    Identifier "screenname"
    Device     "devname"
    Monitor    "Monitor0"
    DefaultDepth24
        Subsection "Display"
            Depth 24
            Modes "1280x1024" "1024x768" "800x600" "640x480"
        EndSubSection
    EndSection

```

8.4.3.3 Monitor Section

Use the Monitor section to define monitor characteristics and timings for a display. You should have one Monitor section for each display your system supports. The Monitor section is referenced in a Screen section. And has the following format.

```

Section "Monitor"
    Identifier "Monitor0"
    VendorName "NEC"
    MonitorName "MEC MultiSync LCD"
    HorizSync  30-60
    VertRefresh 50-75
    ...
EndSection

```

8.4.3.4 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). You typically only need to edit this section when you are using a DIH configuration. You need to add a line to reference the second Screen section and specify its relative location to the first screen. In the following sample, the line beginning with Screen 1 . . . is required for DIH configurations.

```

Section "ServerLayout"
    Identifier "Default Layout"
    Screen 0 "Screen0" 0 0
    Screen 1 "Screen1" RightOf "Screen0"
    InputDevice entries...
EndSection

```

8.4.3.5 ServerFlags Section

If you are configuring the IEGD for Xinerama support, you must set the "Xinerama" option to "True" in the ServerFlags section of the configuration file.

```

Section "ServerFlags"
    Option "Xinerama" "True"
    other options....
EndSection

```

8.4.4 XFree86 and Xorg Configuration Options

Table 21 shows the supported driver options:

Table 21. Supported Driver Options (Sheet 1 of 3)

Option	Description
Option "NoAccel" "boolean"	Disable or enable hardware 2D acceleration. The default is to enable hardware acceleration, so to disable acceleration, set this option to "Yes" or "True" or "1".
Option "SWCursor" "boolean"	Enable the use of the software cursor. The default is off and the hardware cursor is used.
Option "ShadowFB" "boolean"	Enable or disable double buffering on the frame buffer. The default is that double buffering is disabled.
Option "TearFB" "boolean"	Disable or enable wait for vblank when doing blits. The default is to not wait for vblank when doing blits. This is faster but may cause visible tearing of the display.
Option "XVideo" "boolean"	Disable or enable XVideo support. In a dual independent head configuration, either the first display or the second display support XVideo. Both displays can not support XVideo simultaneously. The default is XVideo support is enabled.
Option "XVideoBlend" "boolean"	Disable or enable 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.
Option "VideoKey" "integer"	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.
Option "CloneDisplay" "boolean"	Enable or disable additional output ports on this device. For example, if the graphic device has both an analog and DVO output device, enabling this option will cause both ports to be enabled and the same frame buffer displayed on both. Each port will use a separate graphics pipe so the video timings can be different. See the options below for setting the clones timings. The clone display cannot be set to a larger display mode than the frame buffer. Setting this option will use more video bandwidth than the TwinDisplay option below. Enabling multiple screens takes precedence over this option. The default is CloneDisplay disabled.
Option "CloneWidth" "integer"	This sets the display width for a clone port when CloneDisplay is active. The default is 640.
Option "CloneHeight" "integer"	This sets the display height for a clone port when CloneDisplay is active. The default is 480.
Option "CloneRefresh" "integer"	This sets the display vertical refresh rate for a clone port when CloneDisplay is active. The default is 60Hz.

Table 21. Supported Driver Options (Sheet 2 of 3)

Option	Description
Option "TwinDisplay" "boolean"	Enable or disable additional output ports on this device. For example, if the graphic device has both an analog and DVO output device, enabling this option will cause both ports to be enabled on the same pipe and use the same frame buffer for the display. The video timings will be restricted to modes that both devices support and the devices cannot be programmed with different timings. Enabling CloneDisplay or multiple screens takes precedence over this option. The default is TwinDisplay enabled.
Option "PortDrivers" "string"	<p>This option specifies which port driver(s) must be loaded. The list is a space- or comma-separated list of port driver names corresponding to the *.so port driver files included with the Linux version of the driver. You may specify multiple port driver names if there are multiple DVO devices that are currently on the system or may potentially be used. For example, specify "th164 sii164" to load port drivers for Thine 164 and Silicon Image 164 DVO devices.</p> <p>By default no port driver is loaded except analog port driver for GMCH ANALOG port. This includes the internal LVDS port on the 852, 855, and 915 chipsets. You must at least specify "lvds" to enable use of the internal LVDS port.</p> <p>NOTE: The current implementation of the <code>pcf2iegd</code> utility will not generate this setting properly, so it must be manually added to the configuration file.</p>
Option "PortOrder" "string"	<p>This option can be used to change the default port allocation order. The default order can vary depending on chipset. List the port type numbers in the allocation order starting from first to last. The port type numbers are as follows:</p> <ul style="list-style-type: none"> 1 - Integrated TV Encoder (915GM only) 2 - DVO/sDVO B port 3 - DVO/sDVO C port (or DVO port on 815/E) 4 - Integrated LVDS port (852GME, 855GME, or 915GM only) 5 - Analog CRT port <p>To set the order as Integrated TV Encoder, ANALOG, LVDS, DVO C, DVO B set the PortOption string to "15432". Zeros can be used to specify don't care. Setting this option incorrectly can result in port allocation failures.</p> <p>Note that when specifying the PortOrder value in the .pcf file, do not use double quotes around the value. The <code>pcf2iegd</code> utility automatically inserts double quotes around the value for Linux configurations so that if you specify them in the .pcf file, they will appear as double double-quotes (for example, "PortOrder" ""5421"""). If this occurs, remove one set of quotes from the value ("5421").</p>
Option "PortCtl" "boolean"	Disable or enable the Port Driver Control extension support. When enabled, this extension allows applications to control individual output port attributes and to turn the display port on or off. Note that this option must be enabled in order for the runtime configuration GUI application (<code>IEGDGUI</code>) to work.
Option "AttributePath" "string"	Specify the directory path where attribute data will be saved and restored from. If this is not set, then attribute data will not be saved/restored. The default is no directory path.
Option "Active" "boolean"	Make the device ports active and enabled on startup. This is the default. When this option is set to FALSE, the driver will allocate the output ports for this device but will not set the mode or enable the output. To enable the output, the port control extension must be used.

Table 21. Supported Driver Options (Sheet 3 of 3)

Option	Description
Option "EDID" "boolean"	Enable or disable reading of EDID data from the output port device. Note that if the EDID option is specified in both the config file (XF86Config or Xorg.conf) and the .pcf file, all per-port EDID options in the .pcf file are overwritten by the EDID option specified in the config file.
Option "Rotation" "integer"	Rotate the display. Valid values are 0, 90, 180, 270.
Option "Flip" "boolean"	Invert the display horizontally.

8.4.5 Sample Dual Independent Head (DIH) Configuration

For dual independent head operation, several additional options must be set in the Device sections for each head. Both Device sections must specify the BusID, and the BusID must be the same for both devices. Using different BusID values will cause X to use separate MMIO address ranges for each device. This mode is not supported by this driver. Each Device section must specify the Screen section that will be associated with the device with the Screen option.

```
BusID - B:F:S (Bus, Function, Slot)
Screen - number
```

Figure 25 show a sample DIH configuration. Only the Device, Screen, and Server Layout sections of the configuration file are shown. For a complete example of a configuration file, see Figure 24 on page 132.

Figure 25. Sample DIH Configuration

```
Section "Device"
    Identifier "IntelEGD-1"
    Driver     "intel"
    BusID      "0:2:0"
    Screen     0
    VideoRam   32768
EndSection

Section "Device"
    Identifier "IntelEGD-2"
    Driver     "intel"
    BusID      "0:2:0"
    Screen     1
    VideoRam   32768
    Option     "XVideo"          "No"
EndSection

Section "Screen"
    Identifier "Screen 1"
    Device     "IntelEGD-1"
    Monitor    "Monitor1"
    DefaultDepth 24

    Subsection "Display"
        Depth      8
        Modes       "1280x1024" "1024x768" "800x600" "640x480"
        ViewPort    0 0
    EndSubsection
    Subsection "Display"
        Depth      16
        Modes       "1280x1024" "1024x768" "800x600" "640x480"
        ViewPort    0 0
    EndSubsection
    Subsection "Display"
        Depth      24
        Modes       "1280x1024" "1024x768" "800x600" "640x480"
```

```

        ViewPort      0 0
    EndSubsection EndSection

Section "Screen"
    Identifier "Screen 2"
    Device     "IntelEGD-2"
    Monitor    "Monitor2"
    DefaultDepth 24
    Subsection "Display"
        Depth      8
        Modes       "1280x1024" "1024x768" "800x600" "640x480"
        ViewPort    0 0
    EndSubsection
    Subsection "Display"
        Depth      16
        Modes       "1280x1024" "1024x768" "800x600" "640x480"
        ViewPort    0 0
    EndSubsection
    Subsection "Display"
        Depth      24
        Modes       "1280x1024" "1024x768" "800x600" "640x480"
        ViewPort    0 0
    EndSubsection
EndSection

Section "ServerLayout"
    Identifier "Dual Head Layout"
    Screen "Screen 1"
    Screen "Screen 2" Right Of "Screen 1"
    InputDevice "Mouse1" "CorePointer"
    InputDevice "Keyboard1" "CoreKeyboard"
EndSection

```

8.4.6 Video Memory Management

The Intel 8xx family of integrated graphics chipsets has a unified memory architecture and uses system memory for video RAM. The amount of memory used is not fixed and depends on the configuration. Address space for frame buffers and back buffers is always reserved, along with some scratch space for 2D and 3D acceleration. Use of some features such as Video Overlay also affects video memory allocations. The amount of video memory allocated for pixmap surfaces is configurable. By default, 16 Mbytes is used for each screen. You can change this with the VideoRam entry in the Device section of the configuration file. It may be set to any reasonable value up to 32 Mbytes. Increasing this value reduces the amount of system memory available for other applications. This value is in units of 1024 Kbytes (32 Mbytes is represented by 32768).

8.4.7 Graphics Port Initialization

When used with a graphic chipset that supports multiple graphics pipelines, the driver supports multiple screens and Xinerama. This support is enabled by creating additional Device sections for each additional graphics device on the PCI bus. The driver locates the first device on the bus and associates it with the device section that matches (or one that doesn't specify a busID). This becomes the primary display. If the graphics chipset supports multiple plans and pipes, and the config file specifies two Device sections and two Screen sections, the driver attempts to operate in a dual-independent head mode. Once all the graphics devices and device sections have been matched up, the driver attempts to allocate any remaining output ports and attach them to the primary graphics device. For example:

- Two pipes and two ports allows for dual independent displays.
- One pipe and two ports allows for a cloned display.
- Two pipes and three ports allows for dual independent displays with the third port providing a clone of the first display.

8.4.8 EDID-Less Bin Configuration File

This driver supports the reading of EDID parameter blocks from an external file. To use this feature, you must create and save an EDID-`<config_id>.bin` parameter block file in the `/etc/X11` directory, where `<config_id>` is the configuration ID that is either set via a BIOS setting or read from the `XF86Config` or `Xorg.conf` file. The BIOS setting takes precedence over the `XF86Config` or `Xorg.conf` file setting. If no configuration ID is set, the driver looks for and loads `EDID-0.bin`. If a suitable file is found in the `/etc/X11` directory, it will be read and the EDID parameters will be used by the driver. To disable the reading of the file, remove the file from `/etc/X11`.

In addition to the EDID file, changes must be made to `XF86Config` or `Xorg.conf` file to enable new, nonstandard modes. Adding the DTD and FP_INFO parameter blocks to the EDID file adds the mode(s) timings to the driver list but won't add them to the X-server's default user list. To add new timings to the X server's list, add the mode's name to the appropriate Screen section of the config file.

Note: Modelines are not a supported method in IEGD for configuring display timings. The EDID file must be used. This file serves a similar but more advanced purpose as the ModeLine lines in the `XF86Config` and `Xorg.conf` files.

Typical uses for the EDID configuration file are:

1. Adding user-defined display timings for a display device.
2. Allowing the driver to support non-standard flat panel configurations.
3. Setting flat panel backlight control

8.4.8.1 EDID File Format

The EDID-`<config_id>.bin` file is a binary format. It can be created from a Platform Configuration File (PCF) using the `pcf2iegd -bin` utility. Numeric data in the files are stored in 32 bit little-endian format.

The first 4 bytes of the file are the version number. The current file version is 0x00000400.

The second 4 bytes are the length of the comment string. Following this is the comment string. The string is null-terminated ('\0').

After the comment string there are 5 display port configuration blocks. The block size is determined by the number of timing descriptors and attributes associated with each block.

Port configuration block:

SIZE	TYPE	RANGE	(DESCRIPTION)
4 bytes	unsigned long	[1..5]	(hardware port number)
4 bytes	unsigned long		(data present bits)
4 bytes	unsigned long		(display flags)
2 bytes	unsigned short		(Flags when EDID is available)
		BIT0: 0	- Do not use driver built-in standard timings
		1	- Use driver built-in standard timings
		BIT1: 0	- Do not use EDID block
		1	- Use EDID block and filter modes
		BIT2: 0	- Do not use user-DTDs
		1	- Use user-DTDs.
2 bytes	unsigned short		(Flags when EDID is not available)
		BIT0: 0	- Do not use driver built-in standard timings

```

1 - Use driver built-in standard timings
BIT2: 0 - Do not use user-DTDs
1 - Use user-DTDs.
4 bytes unsigned long [0..6] (DDC GPIO pin pair)
4 bytes unsigned long [0..4] (DDC GPIO speed in kHz)
4 bytes unsigned long (DDC GPIO Device Address byte)
4 bytes unsigned long [0..6] (I2C GPIO pin pair)
4 bytes unsigned long [0..4] (I2C GPIO speed in kHz)
4 bytes unsigned long (I2C GPIO Device Address byte)
4 bytes unsigned long (flat panel width, pixels)
4 bytes unsigned long (flat panel height, pixels)
4 bytes unsigned long [0..3] (flat panel power method,
0 - No method for controlling the power)
1 - Port Driver controls the power)
2 - GMCH GPIO pins control the power)
3 - ICH GPIO pins control the power)
4 bytes unsigned long (T1 - flat panel minimum time delay between
VDD active and DVO clock/data active in
milliseconds)
4 bytes unsigned long (T2 - flat panel minimum time delay between
DVO clock/data active and backlight enable
in milliseconds)
4 bytes unsigned long (T3 - flat panel minimum time delay between
backlight disable and DVO clock/data
inactive in milliseconds)
4 bytes unsigned long (T4 - flat panel minimum time delay between
DVO clock/data inactive and VDD inactive in
milliseconds)
4 bytes unsigned long (T5 - flat panel minimum time delay between
VDD inactive and VDD active in milliseconds)
4 bytes unsigned long (flat panel GPIO pin for controlling the Vdd
power)
4 bytes unsigned long (flat panel GPIO pin for controlling the Vee
backlight power)
4 bytes unsigned long (flat panel GPIO pin for enabling the
backlight)
4 bytes unsigned long (number of Display Timing Descriptors)
4 bytes unsigned long (Reserved)
4 bytes unsigned long (number of attributes)
4 bytes unsigned long (Reserved)

```

Following this are all the Display Timing Descriptors. There can be 0 or more Timing Descriptor blocks. The actual number of blocks is defined by the value above.

```

2 bytes unsigned short (width)
2 bytes unsigned short (height)
2 bytes unsigned short (refresh)
4 bytes unsigned long (dot clock in kHz)
2 bytes unsigned short (x offset)
2 bytes unsigned short (y offset)
2 bytes unsigned short (h_sync start)
2 bytes unsigned short (h_sync length)
2 bytes unsigned short (h_blank start)
2 bytes unsigned short (h_blank length)
2 bytes unsigned short (v_sync start)
2 bytes unsigned short (v_sync length)
2 bytes unsigned short (v_blank start)
2 bytes unsigned short (v_blank length)
2 bytes unsigned short (Reserved)
2 bytes unsigned short (Reserved)
4 bytes unsigned long (flags)

```

Following this are all the attributes. There can be 0 or more attributes. The actual number of attributes is defined by the attribute count value above.

```

4 bytes unsigned long (id - see igd_pd.h)
4 bytes unsigned long (value)

```

Data Present bits

```

DDC_GPIO      0x00000001
DDC_SPEED     0x00000002
DDC_DAB       0x00000004

```

```

I2C_GPIO           0x00000008
I2C_SPEED          0x00000010
I2C_DAB            0x00000020
FLAT_PANEL_INFO    0x00000040
DTD_LIST           0x00000080
ATTRIBUTE_LIST     0x00000100

```

Display Flags

```

READ_MONITOR_EDID  0x00000001
MULTIPLE_DVO       0x00000002

```

Hardware Port Numbers

The hardware port numbers refer to the actual GMCH hardware display port. Which ports actually exist vary depending on the chipset.

```

port 1 - Integrated TV Encoder (915GM only)
port 2 - DVO/sDVO B port
port 3 - DVO/sDVO C port (or DVO port on 815/E)
port 4 - Integrated LVDS port (852GME, 855GME, or 915GM only)
port 5 - Analog CRT port

```

8.4.9 Sample Advanced EDID Configurations for Linux OS

The `edid_avail` and `edid_not_avail` PCF parameters control the available timings for any display. The `edid_avail` parameter is used when EDID information is read from the display. If the driver is unable to read EDID information from the display or if the `edid` parameter in the `.pcf` file is set to "0" (disable), then the settings of the `edid_not_avail` parameter are used. Refer to [Section 3.8, "Advanced EDID Configuration" on page 50](#) for details on specifying these parameters in the PCF file.

There is an `edid` option that can be placed in the `XF86Config` or `Xorg.conf` files that controls the behavior of the overall driver, and there are also EDID settings within the PCF file that control the behavior on each port (`edid`, `edid_avail`, and `edid_not_avail`). The combination of these settings determine how the driver behaves. [Table 22](#) shows various configurations and the expected behavior of the driver.

Table 22. Sample Advanced EDID Configurations for Linux OS

Case	XF86Config "edid" option	PCF: Per port "edid" option	Expected driver behavior
1.	No "edid" option specified	No edid flag specified	For every port, driver uses <code>edid_avail</code> .
2.	No "edid" option specified	<code>edid=0</code> for some ports and <code>edid=1</code> for some ports	For <code>edid=0</code> ports, driver uses <code>edid_not_avail</code> flags. For <code>edid=1</code> ports driver uses <code>edid_avail</code> flags.
3.	"edid"=no	Setting does not matter.	For all ports driver will not read <code>edid</code> and interprets <code>edid_not_avail</code> flags. Driver overrides any per-port pcf <code>edid</code> flags, treats all displays as EDID-less displays, and uses <code>edid_not_avail</code> flags.
4.	"edid"=yes	<code>edid=0</code> for some ports and <code>edid=1</code> for some ports	Same as case 2

Note: For all cases:

1. If there is not an `edid_not_avail` flag specified for a port, and an EDID-less display is detected, then the driver defaults to using the standard built-in timings for that port.
2. If there is not an `edid_avail` flag specified for a port, and an EDID display is detected, then the driver defaults to using the EDID data from the display, plus any user specified DTDs.
3. If `edid=1` and the display device is EDID-less, then the driver uses `edid_not_avail` flags.

8.5 Run Time Operation

8.5.1 Runtime Configuration GUI (IEGDGUI)

You can change the configuration and runtime attributes of the driver using the IEGDGUI runtime configuration GUI. The IEGDGUI resides in the `IEGD_5_0/Utilities` directory. The Intel Embedded Graphics Drivers GUI (IEGDGUI) is a GUI application that is used to view and control the Intel Embedded Graphics Drivers. This tool is used to retrieve status information of the display and driver and also to configure the supported display attributes. The IEGDGUI also demonstrates multi-monitor support.

8.5.1.1 IEGDGUI Setup

In order to run the IEGDGUI, you need to ensure the XServer has been configured to use the IEGD. See [Section 8.4, “Configuration” on page 130](#) for details on configuring and installing the IEGD.

You also need to install the QT library. This library comes freely with the Red Hat* 9.0 Linux distribution or Fedora Core 2 and should already be installed. The version of the QT library for Red Hat 9 is 3.1 and for Fedora Core 2, the version is 3.3. If it is not installed or available, the library can be downloaded from the following website:

www.trolltech.com

To setup the IEGDGUI, follow this procedure:

1. If the library is not installed, obtain it and copy it to the following directory:

```
/usr/lib/qt-3.1/lib (for Red Hat 9)
/usr/lib/qt-3.3/lib (for Fedora Core 2)
```

2. Ensure that the `LD_LIBRARY_PATH` environment variable points to the QT library and the X11R6 library. If it does not, type the following command:

```
export LD_LIBRARY_PATH=/usr/X11R6/lib:/usr/lib/qt-3.1/lib (for Red Hat 9)
export LD_LIBRARY_PATH=/usr/X11R6/lib:/usr/lib/qt-3.3/lib (for Fedora Core 2)
```

3. Create the following symbolic links in the appropriate QT library directory :

For Red Hat 9, create the links in the `/usr/lib/qt-3.1/lib` directory:

```
cd /usr/lib/qt-3.1/lib
ln -s libqt-mt.so.3.1.1 libqt-mt.so.3
ln -s libqt-mt.so.3.1.1 libqt-mt.so.3.1
ln -s libqt-mt.so.3.1.1 libqt.so
ln -s libqt-mt.so.3.1.1 libqt.so.3
ln -s libqt-mt.so.3.1.1 libqt.so.3.1
ln -s libqt-mt.so.3.1.1 libqt.so.3.1.1
```

For Fedora Core 2, create the links in the `/usr/lib/qt-3.3/lib` directory:

```
cd /usr/lib/qt-3.3/lib
ln -s libqt-mt.so libqt.so
ln -s libqt-mt.so.3 libqt-mt.so.3
ln -s libqt-mt.so.3.3 libqt-mt.so.3.3
ln -s libqt-mt.so.3.3.2 libqt.so.3.3.2
```

4. Ensure the IEGDGUI is executable by changing directories to `.../IEGD_5_0_Linux/Utilities` and running the following command:

```
ls -l IEGDGUI
```

Executable permissions should be set for all three Linux groups (root, group, user) and should look like this:

```
rwxr_xr_x IEGDGUI ...
```

If the permissions do not contain an “x” for each group, change the permissions using the following command:

```
chmod +rwx IEGDGUI
```

Once you have completed these steps, the IEGD can be launched.

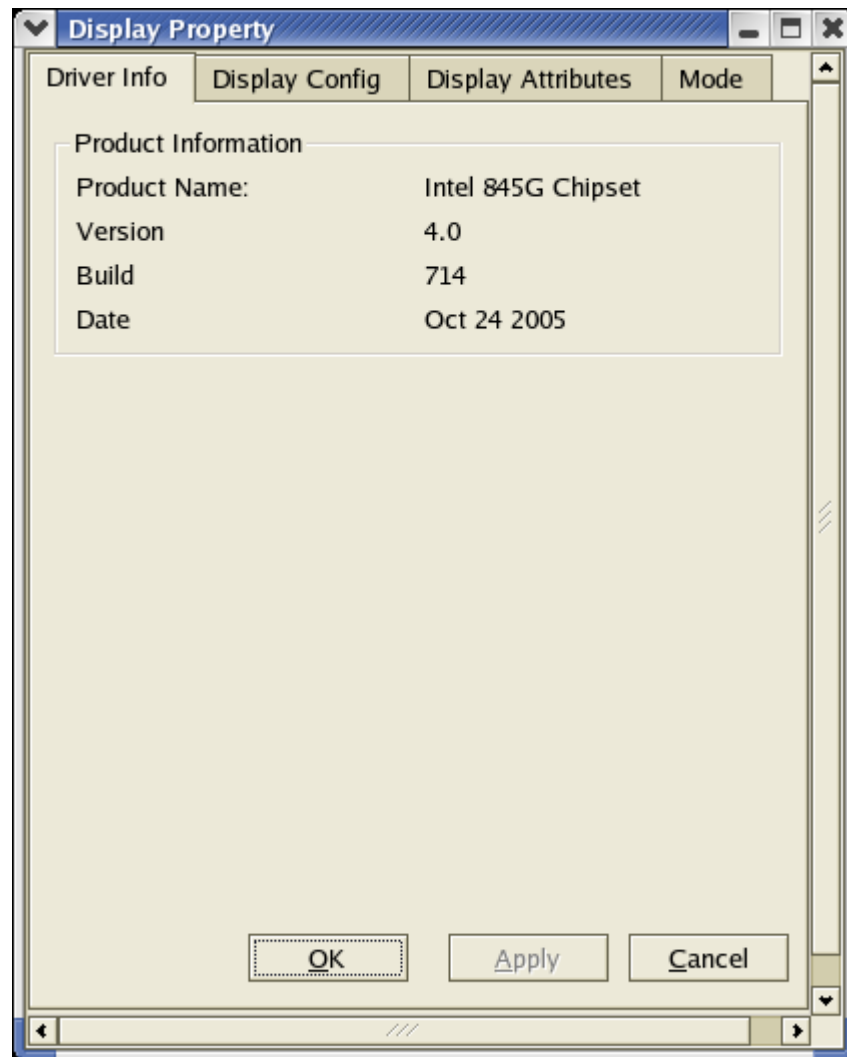
8.5.1.2 Using the IEGDGUI Runtime Configuration Utility

The IEGDGUI application provides three tabs: **Driver Info**, **Display Config**, and **Display Attributes**.

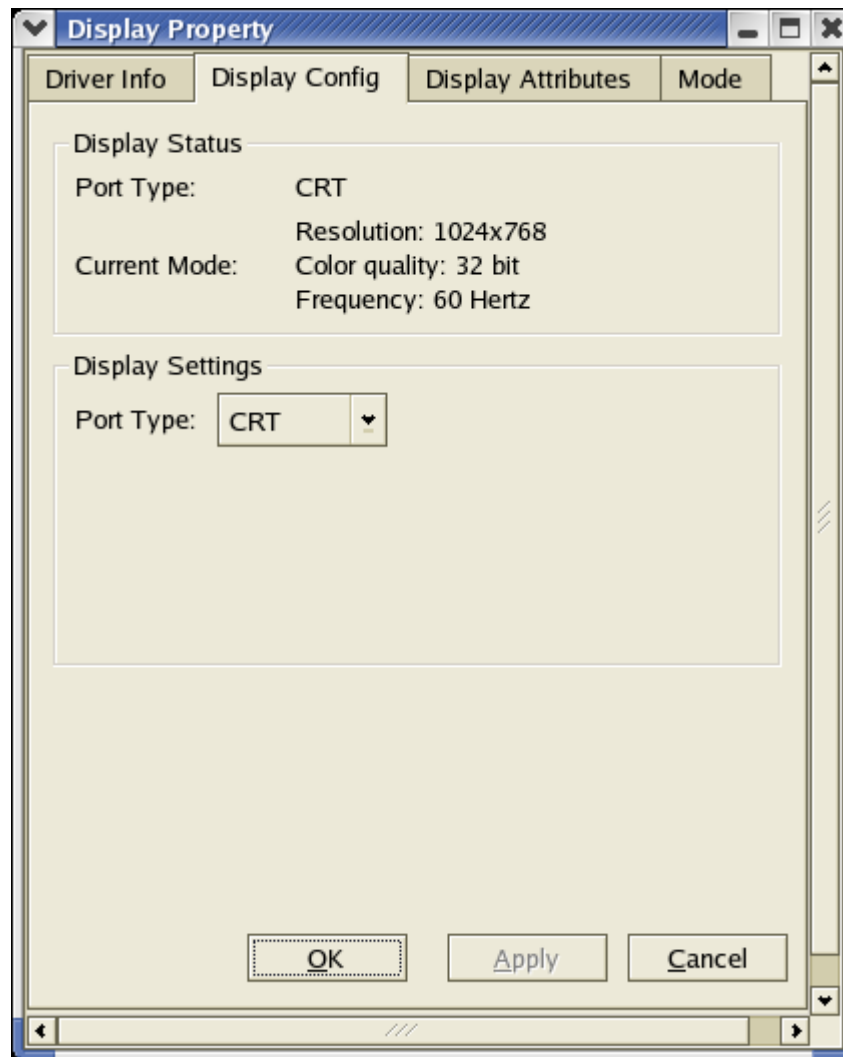
- **Driver Info:** Contains the driver status and display information.
- **Display Config:** Contains current display information and allows configuration of display modes, flip, rotation, and enabling/disabling of a specified port.
- **Display Attributes:** Contains the supported Port Driver (PD) attributes and allows configuration of PD attributes.
- **Mode --** Contains current resolution and bit depth settings for the primary and secondary ports. You can change the settings for both the primary and secondary ports from this tab.

Figure 26 shows the **Driver info** tab.

Figure 26. Linux Runtime Configuration GUI-- Driver Info Tab

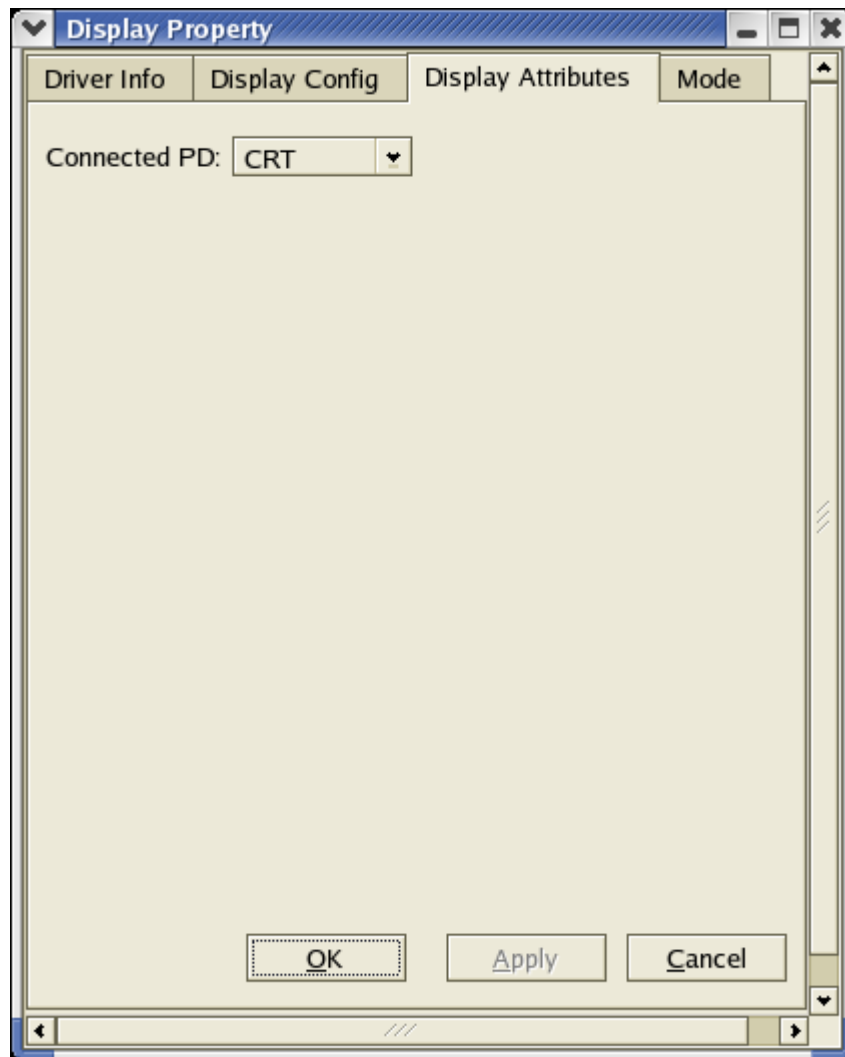


To view current display information and or to change the current configuration of display modes, flip, rotation, and enabling/disabling of a specified port, click the **Display Config** tab. If you make any changes to the configuration, click the **Apply** button for the changes to take effect. [Figure 27](#) shows a sample configuration.

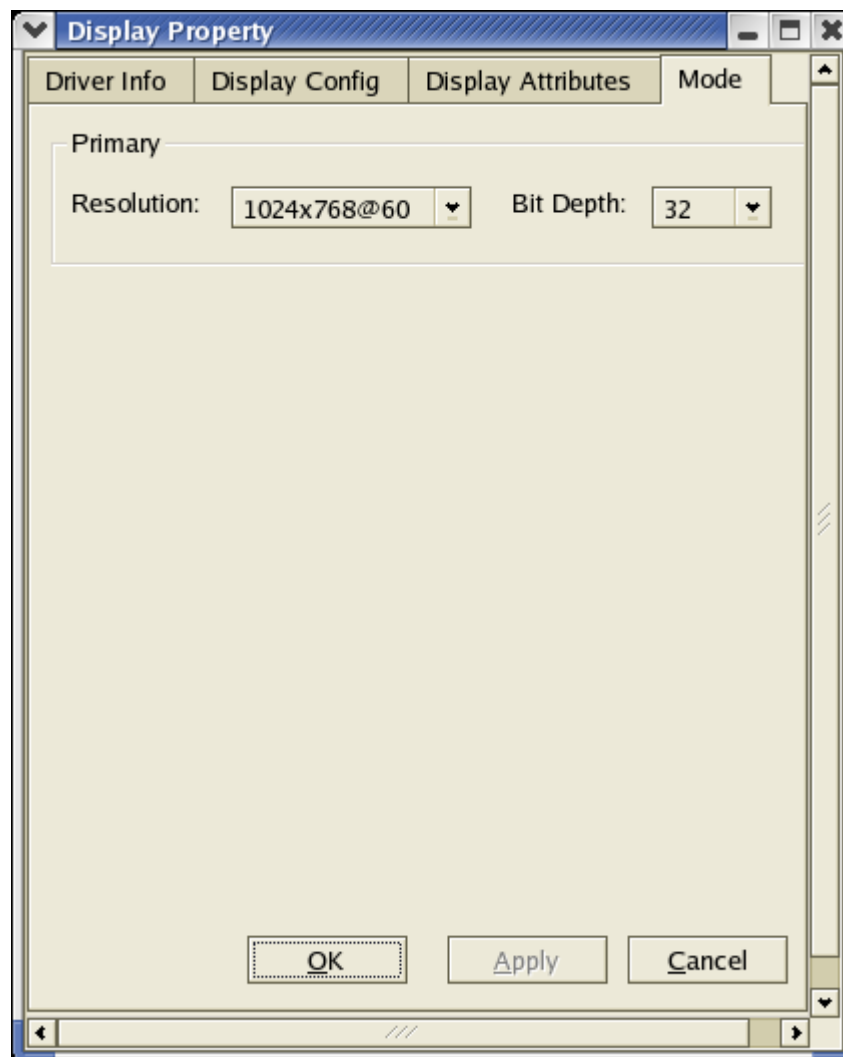
Figure 27. Linux Runtime Configuration GUI -- Display Config Tab

To view or change the current configuration of the Port Driver, click the **Display Attributes** tab. [Figure 28](#) shows a sample configuration. Refer to [Appendix C](#) for detailed information on port driver attributes. If you make any changes to the port driver attributes, click the **Apply** button for the changes to take effect.

Figure 28. Linux Runtime Configuration GUI -- Display Attributes Tab



To view or change the current mode's resolution and bit depths, click the **Mode** tab. The screen that appears depends upon your current configuration. [Figure 29](#) shows a sample Mode tab screen.

Figure 29. Linux Runtime Configuration GUI -- Mode Tab

PCF File Syntax

A

This Appendix describes the complete syntax of the PCF file. It can be used as a template to build a PCF file. For a sample pcf file, refer to [Section 3.6, “Sample PCF File”](#) on page 42.

```
#-----
# PCF Syntax
#-----
# General format rules

# 1. Blocks: Syntax: [[block_name] [block_id]] { }.

#   Blocks are denoted by open and close braces - { }. Blocks may have an
#   optional block_name. block_name's may contain an optional block_id.
#   Blocks may contain sub blocks. Blocks may contain assignments,
#   comments, or blank lines.

#       a. block_name: Non-quoted ascii string
#       b. block_id: Integer

# 2. Assignments: Syntax: <name> = <value>
#       a. name: Non-quoted ascii string
#       b. value: Integer or Quoted ascii string

# 3. Known Vocabulary:
#   comment
#   dtd
#   general
#   portorder
#   port
#   displayconfig
#   configid
#   config
#   rotation
#   displayflags
#   multidvo
#   edid
#   edid_avail
#   edid_not_avail
#   fpinfo
#   width
#   height
#   bkltenable
#   bkltmethod
#   bkltt1
#   bkltt2
#   bkltt3
#   bkltt4
#   bkltt5
#   gpiopinvd
#   gpiopinvee
#   gpiopinbklt
#   dvo
#   i2cpin
#   i2cdab
#   i2cspeed
#   ddcpin
#   ddcspeed
```

```

#      attr
#      p_clock
#      h_active
#      h_syncp
#      h_sync
#      h_border
#      h_blank
#      v_active
#      v_syncp
#      v_sync
#      v_border
#      v_blank
#      h_size
#      v_size
#      flags

# 4. Tokens not found in vocabulary are legal, but they will not be processed
#      into Hungarian format.
# 5. More than 1 DTD is valid within a port block
# 6. More than 1 configuration is valid (1-15)
# 7. Comments can be inserted anywhere in a PCF file. Comment syntax
#      requires the first non-white space character to be a pound sign ('#').
#      All characters following the '#' will then be ignored by the PCF parser.
#
#-----
# Keyword value definitions
#-----
# 1. config_id: Integer. Range [1-0x7fffffff].
# 2. comment_string: A quoted string used to identify the origin of the .bin
#      or .inf file.
# 3. config_name_string: A quoted string used to identify the configuration
#      name.
# 4. display_config: Integer. Range:
#      1 - Single
#      2 - Clone
#      4 - Twin
#      8 - Extended
# 5. portorder: 4 digit decimal Integer. Digits are used left to right.
#      1234 = Port 1 -> Port 2 -> Port 3 -> Port 4
#      This order decides the primary and secondary displays.
#      Example: On i845/i85x systems, portorder = "5342" means port 5 is
#      primary display and port 3 is secondary display and so on.
#      Check user guide for more information.
# 6. port_name_string: A quoted string used to identify the port
# 7. rotation: Integer.
#      Windows Range:
#      0x000 - 0 degrees
#      0x100 - 90 degrees
#      0x200 - 180 degrees
#      0x300 - 270 degrees
#      0x400 - 0 degree rotation with horizontal flip
#      0x500 - 90 degree rotation with horizontal flip
#      0x600 - 180 degree rotation with horizontal flip
#      0x700 - 270 degree rotation with horizontal flip
#      Linux Range:
#      0 - 0 degrees
#      90 - 90 degrees
#      180 - 180 degrees
#      270 - 270 degrees
#
# 8. edid: Boolean [0 | 1]
#      0 - Do not read EDID
#      1 - Read EDID block
#
#      If no 'edid' is specified in the pcf, then driver reads EDID and
#      filter modes.

```

```
#
# 9. edid_avail and edid_not_avail: <16 bits>
# -----
# These 2 parameters can be used to control the available timings for
# any display. 'edid_avail' is used when EDID is read from the display
# device. If the attempt to read EDID is failed or 'edid = 0' then
# driver uses 'edid_not_avail' flags.
#
# See below bit definitions for both 'edid_avail' and 'edid_not_avail'
#
#
# BIT 0:
# -----
# 0 - Do not use driver built-in standard timings
# 1 - Use driver built-in standard timings
#
# BIT1: <not applicable to edid_not_avail>
# -----
# 0 - Do not use EDID block
# 1 - Use EDID block and filter modes
#
# BIT2:
# -----
# 0 - Do not use user-DTDs
# 1 - Use user-DTDs.
#
# BIT3-BIT15
# -----
# Future use.
#
# Default behavior:
# -----
# If user doesn't provide edid_avail and edid_not_avail, then
# edid_avail = Use Std timings + Use EDID block and Filter modes
# edid_not_avail = Use Std timings
#
# 10. i2c_speed: Integer. Range [1-1000]. Units in KHz
# 11. fp_width: Integer. Screen resolution width (required for ns2501)
# 12. fp_height: Integer. Screen resolution height (required for ns2501)
# 13. bklt_method: Integer. Range [0-3]
# 0 - no backlight
# 1 - Port Driver
# 2 - GMCH
# 3 - ICH
# 13. bklt_tdelay: Integer. Range [1-0x7fffffff]. Units in ms
# 15. gpio_pin: Integer. Range [0-0x7fffffff].
#
# -----
# DTD - Detailed Timing Descriptor (16-29)
# -----
# 16. pixel_clock_khz: Integer. Range [0-0x7fffffff].
# 17. horizontal_active: Integer. Range [12 bits].
# 18. horizontal_sync: Integer. Range [10 bits].
# 19. horizontal_sync_pulsewidth: Integer. Range [10 bits].
# 20. horizontal_border: Integer. Range [8 bits].
# 21. horizontal_blank: Integer. Range [12 bits].
# 22. vertical_active: Integer. Range [12 bits].
# 23. vertical_sync: Integer. Range [6 bits].
# 24. vertical_sync_pulsewidth: Integer. Range [6 bits].
# 25. vertical_border: Integer. Range [8 bits].
# 26. vertical_blank: Integer. Range [12 bits].
# 27. horizontal_size: Integer. Range [12 bits].
# 28. vertical_size: Integer. Range [12 bits].
# 29. flags: Integer. Range [32 bits].
#
# bit 31
# -----
```

```

#      0 - Non-interlaced
#      1 - Interlaced
#
#      bit 27
#      -----
#      0 - vertical sync polarity active low
#      1 - vertical sync polarity active high
#
#      bit 26
#      -----
#      0 - horizontal sync polarity active low
#      1 - horizontal sync polarity active high
#
#      bit 25
#      -----
#      0 - blank sync polarity active high
#      1 - blank sync polarity active low
#
#      bit 17
#      -----
#      0 - Normal DTD
#      1 - Panel/display Native DTD
#
#      All other bits
#      -----
#      Do not use any other bits and must be 0.
#
# -----
# Attributes
# -----
# 30. attribute_id: Unsigned long.  Range [0-0xffffffff]
#
# -----
# Vbios Block
# -----
# The vBIOS block is only used when creating a Video Bios. Other drivers
# will ignore all vbios block parameters.
#
# common_to_port:
# Mapping for legacy int15 5f35 common names to port numbers. Unused
# names should be mapped to port 0. When the system bios reports a
# display to be off by setting a bit in the 5f35 return, the corresponding
# port will be disabled.
# example
# common_to_port = 503000
#
# oem_string:
# OEM string returned from VBE 00. Default value is provided
#
# oem_vendor_name
# oem_product_name
# oem_product_rev:
# Strings returned by VBE 00. Default is empty string(s)
#
# default_refresh_<0|1>:
# Default refresh rate to use for VBE modes, typically 60, 85 etc.
# Default value is 60hz
#
# default_mode_<0|1>:
# Default Video mode to set during initialization (Used in TSR)
# VGA mode 3 is default
#
# default_vga_height = <200 | 350 | 400> #default is 400
# VGA mode height to use. This provides support for CGA, EGA and VGA
# versions of modes 0-4, and 7 when applicable.
#
# -----

```



```

{
    # pd block and all members are optional.
    pd {
        sii164 = <0 | 1>
        ch7009 = <0 | 1>
        tl955 = <0 | 1>
        ns2501 = <0 | 1>
        th164 = <0 | 1>
        fs454 = <0 | 1>
        ns387 = <0 | 1>
        lvds = <0 | 1>
        ch7017 = <0 | 1>
    }

    config <config_id> {
        comment = "<comment_string>"
        name = "<config_name_string>"
        general {
            displayconfig = <display_config>
            portorder      = "port order string"
        }

        # vbios block and all members are optional.
        vbios {
            common_to_port = 503000
            oem_string = "<string>"
            oem_vendor_name = "<string>"
            oem_product_name = "<string>"
            oem_product_rev = "<string>"

            default_refresh_0 = <integer> #default is 60
            default_refresh_1 = <integer> #default is 60

            default_mode_0 = <integer> #default is 3
            default_mode_1 = <integer> #default is 3

            default_vga_height = <200 | 350 | 400>
            #default is 400
        }

        port <port_id> {
            name = "<port_name_string>"
            general {
                rotation = <rotation>
                edid = <0 | 1>
                edid_avail = <16 bits>
                edid_not_avail = <16 bits>
                multivbo = <0 | 1>
            }
            dvo {
                i2cpin = <0-6>
                i2cdab = <0x00-0xff>
                i2cspeed = <i2c_speed>
                ddcpin = <0-6>
                ddcdab = <0x00-0xff>
                ddcspeed = <i2c_speed>
            }
            fpinfo {
                width = <fp_width>
                height = <fp_height>
                bkltmethod = <bklt_method>
                bkltt1 = <bklt_tdelay>
                bkltt2 = <bklt_tdelay>
                bkltt3 = <bklt_tdelay>
                bkltt4 = <bklt_tdelay>
                bkltt5 = <bklt_tdelay>
            }
        }
    }
}

```

```

        gpiopinvee = <gpio_pin>
        gpiopinvee = <gpio_pin>
        gpiopinbklt = <gpio_pin>
        UseGMCHClockPin =
        <bool_use_clock_pins_vee_dd_bklt>
        UseGMCHDataPin =
        <bool_use_data_pins_vee_dd_bklt>
    }
    dtd <dtd_id> {
        p_clock = <pixel_clock_khz>

        h_active = <horizontal_active>
        h_sync = <horizontal_sync>
        h_syncp = <horizontal_sync_pulsewidth>
        h_border = <horizontal_border>
        h_blank = <horizontal_blank>

        v_active = <vertical_active>
        v_sync = <vertical_sync>
        v_syncp = <vertical_sync_pulsewidth>
        v_border = <vertical_border>
        v_blank = <vertical_blank>

        h_size = <horizontal_size>
        v_size = <vertical_size>

        flags = <dtd_flags>
    }
    attr {
        id <attribute_id> = <0-2^32>
    }
}
}
}
}

```

Example INF File

B

```

;*****
; Filename: ssigd.inf
; $Revision: 1.434 $
; $Id: ssigd.inf,v 1.434 2005/09/08 05:13:46 anthill Exp $
; $Source: /nfs/fm/proj/eia/cvsroot/ssigd/ial/dx/install/ssigd.inf,v $
;
; Copyright (c) 2003 Intel Corporation. All rights reserved.
;
;*****

[Version]
Signature="$WINDOWS NT$"
Class=Display
ClassGUID={4D36E968-E325-11CE-BFC1-08002BE10318}
Provider=%Intel%
CatalogFile=ssigd.cat
DriverVer=09/07/2005,4.0.0.691

;=====
[SourceDisksNames]
1=%DiskDesc%,,""

[SourceDisksFiles]
igdmini.sys = 1
igddis.dll = 1
igd3dalm.dll= 1
d3dUtil.sys = 1
ch7009.sys = 1
ch7017.sys = 1
cx873.sys = 1
fs454.sys = 1
fs460.sys = 1
lvds.sys = 1
ns2501.sys = 1
ns387.sys = 1
siil64.sys = 1
thl64.sys = 1
sdvo.sys = 1

;=====
[DestinationDirs]
DefaultDestDir = 11; System directory
ssigd.Display_wht = 11
ssigd.Display_alm = 11
ssigd.Display_nap = 11
ssigd.Miniport = 12; Drivers directory
ssigd.d3dUtil = 12
ssigd.PortDrvs_wht = 12
ssigd.PortDrvs_alm = 12
ssigd.PortDrvs_nap = 12

;=====
[Manufacturer]
%Intel%=Intel.Mfg

;=====
[Intel.Mfg]
%Intel% %i810% = ssigd_wht, PCI\VEN_8086&DEV_7121
%Intel% %i810DC% = ssigd_wht, PCI\VEN_8086&DEV_7123
%Intel% %i810E% = ssigd_wht, PCI\VEN_8086&DEV_7125
%Intel% %i815% = ssigd_wht, PCI\VEN_8086&DEV_1132

%Intel% %i830m% = ssigd_alm, PCI\VEN_8086&DEV_3577
%Intel% %i835% = ssigd_alm, PCI\VEN_8086&DEV_357B
%Intel% %i845% = ssigd_alm, PCI\VEN_8086&DEV_2562
%Intel% %i855% = ssigd_alm, PCI\VEN_8086&DEV_3582

```

```

%Intel% %i865%      = ssigd_alm, PCI\VEN_8086&DEV_2572

%Intel% %i915GD0% = ssigd_nap, PCI\VEN_8086&DEV_2582
%Intel% %i915GD1% = ssigd_nap, PCI\VEN_8086&DEV_2782
%Intel% %i915AL0% = ssigd_nap, PCI\VEN_8086&DEV_2592
%Intel% %i915AL1% = ssigd_nap, PCI\VEN_8086&DEV_2792
%Intel% %i945LP0% = ssigd_nap, PCI\VEN_8086&DEV_2772
%Intel% %i945LP1% = ssigd_nap, PCI\VEN_8086&DEV_2776
%Intel% %i945CT0% = ssigd_nap, PCI\VEN_8086&DEV_27A2
%Intel% %i945CT1% = ssigd_nap, PCI\VEN_8086&DEV_27A6

;=====
[ssigd_wht.GeneralConfigData]
MaximumNumberOfDevices = 2
MaximumDeviceMemoryConfiguration = 128

[ssigd_alm.GeneralConfigData]
MaximumNumberOfDevices = 2
MaximumDeviceMemoryConfiguration = 128

[ssigd_nap.GeneralConfigData]
MaximumNumberOfDevices = 2
MaximumDeviceMemoryConfiguration = 256

;=====
[ssigd_wht]
CopyFiles = ssigd.Miniport, ssigd.Display_wht, ssigd.D3DUtil, ssigd.PortDrvs_wht

[ssigd_alm]
CopyFiles = ssigd.Miniport, ssigd.Display_alm, ssigd.D3DUtil, ssigd.PortDrvs_alm

[ssigd_nap]
CopyFiles = ssigd.Miniport, ssigd.Display_nap, ssigd.D3DUtil, ssigd.PortDrvs_nap

;=====
[ssigd.Miniport]
igdmini.sys

[ssigd.Display_wht]
igddis.dll

[ssigd.Display_alm]
igddis.dll
igd3dalm.dll

[ssigd.Display_nap]
igddis.dll
igd3dalm.dll

[ssigd.D3DUtil]
d3dutil.sys

[ssigd.PortDrvs_wht]
ch7009.sys
ch7017.sys
fs454.sys
lvds.sys
ns2501.sys
ns387.sys
sii164.sys
th164.sys

[ssigd.PortDrvs_alm]
ch7009.sys
ch7017.sys
fs454.sys
lvds.sys
ns2501.sys
ns387.sys
sii164.sys
th164.sys

[ssigd.PortDrvs_nap]
sdvo.sys
lvds.sys

```

```

;=====
[ssigd_wht.Services]
AddService = igdmini, 0x00000002, ssigd_Service_Inst, ssigd_EventLog_Inst
AddService = ch7009, ,ch7009_Service_Inst, ssigd_EventLog_Inst
AddService = ch7017, ,ch7017_Service_Inst, ssigd_EventLog_Inst
AddService = fs454, ,fs454_Service_Inst, ssigd_EventLog_Inst
AddService = lvds, ,lvds_Service_Inst, ssigd_EventLog_Inst
AddService = ns2501, ,ns2501_Service_Inst, ssigd_EventLog_Inst
AddService = ns387, ,ns387_Service_Inst, ssigd_EventLog_Inst
AddService = sii164, ,sii164_Service_Inst, ssigd_EventLog_Inst
AddService = th164, ,th164_Service_Inst, ssigd_EventLog_Inst

[ssigd_alm.Services]
AddService = igdmini, 0x00000002, ssigd_Service_Inst, ssigd_EventLog_Inst
AddService = d3dUtil, ,d3dUtil_Service_Inst, ssigd_EventLog_Inst
AddService = ch7009, ,ch7009_Service_Inst, ssigd_EventLog_Inst
AddService = ch7017, ,ch7017_Service_Inst, ssigd_EventLog_Inst
AddService = fs454, ,fs454_Service_Inst, ssigd_EventLog_Inst
AddService = lvds, ,lvds_Service_Inst, ssigd_EventLog_Inst
AddService = ns2501, ,ns2501_Service_Inst, ssigd_EventLog_Inst
AddService = ns387, ,ns387_Service_Inst, ssigd_EventLog_Inst
AddService = sii164, ,sii164_Service_Inst, ssigd_EventLog_Inst
AddService = th164, ,th164_Service_Inst, ssigd_EventLog_Inst

[ssigd_nap.Services]
AddService = igdmini, 0x00000002, ssigd_Service_Inst, ssigd_EventLog_Inst
AddService = d3dUtil, ,d3dUtil_Service_Inst, ssigd_EventLog_Inst
AddService = lvds, ,lvds_Service_Inst, ssigd_EventLog_Inst
AddService = sdvo, ,sdvo_Service_Inst, ssigd_EventLog_Inst

;=====
[ssigd_Service_Inst]
ServiceType = 1
StartType = %SERVICE_DEMAND_START%
ErrorControl = 0
LoadOrderGroup = Video
ServiceBinary = %12%\igdmini.sys

[d3dUtil_Service_Inst]
DisplayName = "d3dutil"
ServiceType = %SERVICE_KERNEL_DRIVER%
StartType = %SERVICE_DEMAND_START%
ErrorControl = %SERVICE_ERROR_IGNORE%
ServiceBinary = %12%\d3dutil.sys

[ch7009_Service_Inst]
DisplayName = "ch7009"
ServiceType = %SERVICE_KERNEL_DRIVER%
StartType = %SERVICE_DEMAND_START%
ErrorControl = %SERVICE_ERROR_IGNORE%
ServiceBinary = %12%\ch7009.sys

[ch7017_Service_Inst]
DisplayName = "ch7017"
ServiceType = %SERVICE_KERNEL_DRIVER%
StartType = %SERVICE_DEMAND_START%
ErrorControl = %SERVICE_ERROR_IGNORE%
ServiceBinary = %12%\ch7017.sys

[fs454_Service_Inst]
DisplayName = "fs454"
ServiceType = %SERVICE_KERNEL_DRIVER%
StartType = %SERVICE_DEMAND_START%
ErrorControl = %SERVICE_ERROR_IGNORE%
ServiceBinary = %12%\fs454.sys

[lvds_Service_Inst]
DisplayName = "lvds"
ServiceType = %SERVICE_KERNEL_DRIVER%
StartType = %SERVICE_DEMAND_START%
ErrorControl = %SERVICE_ERROR_IGNORE%
ServiceBinary = %12%\lvds.sys

[ns2501_Service_Inst]

```

```

DisplayName      = "ns2501"
ServiceType      = %SERVICE_KERNEL_DRIVER%
StartType        = %SERVICE_DEMAND_START%
ErrorControl      = %SERVICE_ERROR_IGNORE%
ServiceBinary    = %12%\ns2501.sys

[ns387_Service_Inst]
DisplayName      = "ns387"
ServiceType      = %SERVICE_KERNEL_DRIVER%
StartType        = %SERVICE_DEMAND_START%
ErrorControl      = %SERVICE_ERROR_IGNORE%
ServiceBinary    = %12%\ns387.sys

[sii164_Service_Inst]
DisplayName      = "sii164"
ServiceType      = %SERVICE_KERNEL_DRIVER%
StartType        = %SERVICE_DEMAND_START%
ErrorControl      = %SERVICE_ERROR_IGNORE%
ServiceBinary    = %12%\sii164.sys

[th164_Service_Inst]
DisplayName      = "th164"
ServiceType      = %SERVICE_KERNEL_DRIVER%
StartType        = %SERVICE_DEMAND_START%
ErrorControl      = %SERVICE_ERROR_IGNORE%
ServiceBinary    = %12%\th164.sys

[sdvo_Service_Inst]
DisplayName      = "sdvo"
ServiceType      = %SERVICE_KERNEL_DRIVER%
StartType        = %SERVICE_DEMAND_START%
ErrorControl      = %SERVICE_ERROR_IGNORE%
ServiceBinary    = %12%\sdvo.sys

;=====
[ssigd_EventLog_Inst]
AddReg = ssigd_EventLog_AddReg

[ssigd_EventLog_AddReg]
HKR,,EventMessageFile,0x00020000,"%SystemRoot%\System32\IoLogMsg.dll;%SystemRoot%\System32\drivers\igdmini.sys"
HKR,,TypesSupported,0x00010001,7

;=====
[ssigd_wht.SoftwareSettings]
AddReg = ssigd_SoftwareDeviceSettings_wht

[ssigd_alm.SoftwareSettings]
AddReg = ssigd_SoftwareDeviceSettings_alm

[ssigd_nap.SoftwareSettings]
AddReg = ssigd_SoftwareDeviceSettings_nap

;=====
[ssigd_SoftwareDeviceSettings_wht]
HKR,,InstalledDisplayDrivers,%REG_MULTI_SZ%,igddis
HKR,,MultiFunctionSupported,%REG_MULTI_SZ%,1
HKR,,VgaCompatible,%REG_DWORD%,0
HKR,,PcfVersion,%REG_DWORD%,0x0400

HKR,,No_D3D,%REG_DWORD%,1
HKR,,PortDrivers,%REG_SZ%, "ch7009 ch7017 fs454 lvds ns2501 ns387 sii164 th164"

;-----
[ssigd_SoftwareDeviceSettings_alm]
HKR,,InstalledDisplayDrivers,%REG_MULTI_SZ%,igddis
HKR,,MultiFunctionSupported,%REG_MULTI_SZ%,1
HKR,,VgaCompatible,%REG_DWORD%,0
HKR,,PcfVersion,%REG_DWORD%,0x0400

HKR,,No_D3D,%REG_DWORD%,0
HKR,,PortDrivers,%REG_SZ%, "ch7009 ch7017 fs454 lvds ns2501 ns387 sii164 th164"

;-----
[ssigd_SoftwareDeviceSettings_nap]

```

```

HKR,, InstalledDisplayDrivers, %REG_MULTI_SZ%, igddis
HKR,, MultiFunctionSupported, %REG_MULTI_SZ%, 1
HKR,, VgaCompatible, %REG_DWORD%, 0
HKR,, PcfVersion, %REG_DWORD%, 0x0400

HKR,, No_D3D, %REG_DWORD%, 0

HKR,, PortDrivers, %REG_SZ%, "sdvo lvds"

HKR,, PcfVersion, %REG_DWORD%, 0x0400
HKR, pd, siil64, %REG_DWORD%, 1
HKR, Config\1\General, DisplayConfig, %REG_DWORD%, 4
HKR, Config\1\General, DisplayDetect, %REG_DWORD%, 1
HKR, Config\1\General, PortOrder, %REG_SZ%, "3520"
HKR, Config\1\Port\3\General, Rotation, %REG_DWORD%, 0
HKR, Config\1\Port\3\General, Edid, %REG_DWORD%, 1
HKR, Config\1\Port\3\General, EdidAvail, %REG_DWORD%, 3
HKR, Config\1\Port\3\General, EdidNotAvail, %REG_DWORD%, 1
HKR, Config\1\Port\3\General, MultiDvo, %REG_DWORD%, 1
HKR, Config\1\Port\3\Dvo, DdcPin, %REG_DWORD%, 3
HKR, Config\1\Port\3\Dvo, DdcSpeed, %REG_DWORD%, 100
HKR, Config\1\Port\3\Dvo, ddc dab, %REG_DWORD%, 165
HKR, Config\1\Port\3\Dvo, I2cPin, %REG_DWORD%, 4
HKR, Config\1\Port\3\Dvo, I2cSpeed, %REG_DWORD%, 120
HKR, Config\1\Port\3\Dvo, I2cDab, %REG_DWORD%, 104
HKR, Config\1\Port\3\FpInfo, Width, %REG_DWORD%, 800
HKR, Config\1\Port\3\FpInfo, Height, %REG_DWORD%, 600
HKR, Config\1\Port\3\FpInfo, bklmethod, %REG_DWORD%, 1
HKR, Config\1\Port\3\FpInfo, BkltT1, %REG_DWORD%, 50
HKR, Config\1\Port\3\FpInfo, BkltT2, %REG_DWORD%, 25
HKR, Config\1\Port\3\FpInfo, BkltT3, %REG_DWORD%, 15
HKR, Config\1\Port\3\FpInfo, BkltT4, %REG_DWORD%, 10
HKR, Config\1\Port\3\FpInfo, BkltT5, %REG_DWORD%, 5
HKR, Config\1\Port\3\FpInfo, GpioPinVdd, %REG_DWORD%, 3
HKR, Config\1\Port\3\FpInfo, GpioPinVee, %REG_DWORD%, 5
HKR, Config\1\Port\3\FpInfo, gpiopinenable, %REG_DWORD%, 1
HKR, Config\1\Port\3\Dtd\1, PixelClock, %REG_DWORD%, 30000
HKR, Config\1\Port\3\Dtd\1, HorzActive, %REG_DWORD%, 800
HKR, Config\1\Port\3\Dtd\1, HorzSync, %REG_DWORD%, 32
HKR, Config\1\Port\3\Dtd\1, HorzSyncPulse, %REG_DWORD%, 120
HKR, Config\1\Port\3\Dtd\1, HorzBorder, %REG_DWORD%, 0
HKR, Config\1\Port\3\Dtd\1, HorzBlank, %REG_DWORD%, 224
HKR, Config\1\Port\3\Dtd\1, VertActive, %REG_DWORD%, 600
HKR, Config\1\Port\3\Dtd\1, VertSync, %REG_DWORD%, 3
HKR, Config\1\Port\3\Dtd\1, VertSyncPulse, %REG_DWORD%, 2
HKR, Config\1\Port\3\Dtd\1, VertBorder, %REG_DWORD%, 0
HKR, Config\1\Port\3\Dtd\1, VertBlank, %REG_DWORD%, 24
HKR, Config\1\Port\3\Dtd\1, HorzSize, %REG_DWORD%, 25
HKR, Config\1\Port\3\Dtd\1, VertSize, %REG_DWORD%, 19
HKR, Config\1\Port\3\Dtd\1, Flags, %REG_DWORD%, 0
HKR, Config\1\Port\3\Attr, 2, %REG_DWORD%, 50
HKR, Config\1\Port\3\Attr, 4, %REG_DWORD%, 25
HKR, Config\1\Port\3\Attr, 6, %REG_DWORD%, 75
HKR, Config\1\Port\3\Attr, 8, %REG_DWORD%, 100
HKR, Config\1\Port\3\Attr, 16, %REG_DWORD%, 35
HKR, Config\1\Port\3\Attr, 9, %REG_DWORD%, 45
HKR, Config\2, name, %REG_SZ%, "Intel(R) Embedded Graphics
Driver Platform Configuration File"
HKR, Config\2\General, DisplayConfig, %REG_DWORD%, 2
HKR, Config\2\General, DisplayDetect, %REG_DWORD%, 1
HKR, Config\2\General, PortOrder, %REG_SZ%, "3520"
HKR, Config\2\Port\3, name, %REG_SZ%, "port3"
HKR, Config\2\Port\3\General, Rotation, %REG_DWORD%, 0
HKR, Config\2\Port\3\General, Edid, %REG_DWORD%, 1
HKR, Config\2\Port\3\General, EdidAvail, %REG_DWORD%, 3
HKR, Config\2\Port\3\General, EdidNotAvail, %REG_DWORD%, 1
HKR, Config\2\Port\3\General, MultiDvo, %REG_DWORD%, 1
HKR, Config\2\Port\3\Dvo, DdcPin, %REG_DWORD%, 3
HKR, Config\2\Port\3\Dvo, DdcSpeed, %REG_DWORD%, 100
HKR, Config\2\Port\3\Dvo, ddc dab, %REG_DWORD%, 165
HKR, Config\2\Port\3\Dvo, I2cPin, %REG_DWORD%, 4
HKR, Config\2\Port\3\Dvo, I2cSpeed, %REG_DWORD%, 120
HKR, Config\2\Port\3\Dvo, I2cDab, %REG_DWORD%, 104
HKR, Config\2\Port\3\FpInfo, Width, %REG_DWORD%, 800

```

```

HKR, Config\2\Port\3\FpInfo, Height, %REG_DWORD%, 600
HKR, Config\2\Port\3\FpInfo, bklmethod, %REG_DWORD%, 1
HKR, Config\2\Port\3\FpInfo, BkltT1, %REG_DWORD%, 50
HKR, Config\2\Port\3\FpInfo, BkltT2, %REG_DWORD%, 25
HKR, Config\2\Port\3\FpInfo, BkltT3, %REG_DWORD%, 15
HKR, Config\2\Port\3\FpInfo, BkltT4, %REG_DWORD%, 10
HKR, Config\2\Port\3\FpInfo, BkltT5, %REG_DWORD%, 5
HKR, Config\2\Port\3\FpInfo, GpioPinVdd, %REG_DWORD%, 3
HKR, Config\2\Port\3\FpInfo, GpioPinVee, %REG_DWORD%, 5
HKR, Config\2\Port\3\FpInfo, gpiopinenable, %REG_DWORD%, 1
HKR, Config\2\Port\3\Dtd\2, PixelClock, %REG_DWORD%, 25000
HKR, Config\2\Port\3\Dtd\2, HorzActive, %REG_DWORD%, 640
HKR, Config\2\Port\3\Dtd\2, HorzSync, %REG_DWORD%, 8
HKR, Config\2\Port\3\Dtd\2, HorzSyncPulse, %REG_DWORD%, 128
HKR, Config\2\Port\3\Dtd\2, HorzBorder, %REG_DWORD%, 0
HKR, Config\2\Port\3\Dtd\2, HorzBlank, %REG_DWORD%, 160
HKR, Config\2\Port\3\Dtd\2, VertActive, %REG_DWORD%, 480
HKR, Config\2\Port\3\Dtd\2, VertSync, %REG_DWORD%, 10
HKR, Config\2\Port\3\Dtd\2, VertSyncPulse, %REG_DWORD%, 2
HKR, Config\2\Port\3\Dtd\2, VertBorder, %REG_DWORD%, 0
HKR, Config\2\Port\3\Dtd\2, VertBlank, %REG_DWORD%, 45
HKR, Config\2\Port\3\Dtd\2, HorzSize, %REG_DWORD%, 21
HKR, Config\2\Port\3\Dtd\2, VertSize, %REG_DWORD%, 16
HKR, Config\2\Port\3\Dtd\2, Flags, %REG_DWORD%, 0
HKR, Config\2\Port\3\Attr, 2, %REG_DWORD%, 50
HKR, Config\2\Port\3\Attr, 4, %REG_DWORD%, 25
HKR, Config\2\Port\3\Attr, 8, %REG_DWORD%, 100
HKR, Config\2\Port\3\Attr, 16, %REG_DWORD%, 35

;=====
[Strings]

;-----
; Localizable Strings
;-----
Intel="Intel Corporation"
DiskDesc="Embedded Installation"

i810="810 Embedded Graphics Controller"
i810DC="810DC Embedded Graphics Controller"
i810E="810E Embedded Graphics Controller"
i815="815 Embedded Graphics Controller"

i830m="830M Embedded Graphics Controller"
i835="835 Embedded Graphics Controller"
i845="845 Embedded Graphics Controller"
i855="855 Embedded Graphics Controller"
i865="865 Embedded Graphics Controller"

i915GD0="915G/915GV/910GL Embedded Graphics Controller Function 0"
i915GD1="915G/915GV/910GL Embedded Graphics Controller Function 1"
i915AL0="915GM/915GMS/910GML Embedded Graphics Controller Function 0"
i915AL1="915GM/915GMS/910GML Embedded Graphics Controller Function 1"
i945LP0="945G Embedded Graphics Controller Function 0"
i945LP1="945G Embedded Graphics Controller Function 1"
i945CT0="945GM Embedded Graphics Controller Function 0"
i945CT1="945GM Embedded Graphics Controller Function 1"

;-----
; Non Localizable Strings
;-----
SERVICE_BOOT_START = 0x0
SERVICE_SYSTEM_START = 0x1
SERVICE_AUTO_START = 0x2
SERVICE_DEMAND_START = 0x3
SERVICE_DISABLED = 0x4

SERVICE_KERNEL_DRIVER = 0x1

SERVICE_ERROR_IGNORE = 0x0; Continue on driver load fail
SERVICE_ERROR_NORMAL = 0x1; Display warn, but continue
SERVICE_ERROR_SEVERE = 0x2; Attempt LastKnownGood
SERVICE_ERROR_CRITICAL = 0x3; Attempt LastKnownGood, BugCheck

REG_EXPAND_SZ = 0x00020000

```




Example INF File

```
REG_MULTI_SZ = 0x00010000  
REG_DWORD   = 0x00010001  
REG_SZ      = 0x00000000
```

This page intentionally left blank.

Port Driver Attributes

C

C.1 Standard Port Driver Attributes

Table 23 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 sections of this appendix.

Note: Not all standard attributes are supported by all port drivers. Refer to the following sections for details on the specific attributes supported by each port driver.

Table 23. Standard Port Driver Attributes

Attribute Name	Attribute ID Number	Description
BRIGHTNESS	0	Brightness adjustment.
CONTRAST	1	Contrast adjustment.
HUE	2	Hue adjustment.
FLICKER	3	Setting to reduce flicker.
HPOSITION	4	Controls the horizontal position of the display.
VPOSITION	5	Controls the vertical position of the display.
HSCALE	6	Horizontal scaling ratio.
VSCALE	7	Vertical scaling ratio.
TVFORMAT	8	TV formats are device-specific.
DISPLAY DEVICE	9	Allows selection of different displays for multi-display devices. This attribute is device-specific.
LUMA FILTER	10	TV Luma Filter adjustment.
CHROMA FILTER	11	ChromaFilter adjustment.
TEXT FILTER	12	TV Luma Filter adjustment.
MACROVISION	13	Video Copy Protection.
TV OUTPUT TYPE	14	TV output types. This attribute is device-specific. Note: TV Output Types are limited to svideo and composite for the VBIOS.
SATURATION	15	Saturation adjustment.
FP WIDTH	16	Flat panel width.
FP HEIGHT	17	Flat panel height.
PANEL FIT	18	Panel fitting. Yes or no.
SCALING RATIO	19	Output Scaling. Device-specific.
FP BACKLIGHT ENABLE	20	Enable flat panel backlight.
FP PWR T1	21	VDD Active - DVO CLK/DATA active.

Table 23. Standard Port Driver Attributes

Attribute Name	Attribute ID Number	Description
FP PWR T2	22	DVO active - Backlight enable.
FP PWR T3	23	DVO inactive - Backlight disable.
FP PWR T4	24	DVO inactive - VDD inactive.
FP PWR T5	25	VDD inactive - VDD active
PANEL DEPTH	26	Range 18 - 24.
2 CHANNEL PANEL	27	Dual-channel panel.
GANG MODE	28	Gang display or not.
GANG MODE EVEN ODD	29	Gang display even or odd.
REVERSE DVO DATA	30	Reverses DVO data order.
SHARPNESS	31	Sharpness.
HWCONFIG	32	Hardware Configuration for DVO encoders that support multiple configurations.
HORZFILTER	33	Horizontal Filter.
VERTFILTER	34	Vertical Filter.

C.2 Port Driver Customized Attributes

This section provides information on TV and Flat Panel attributes for specific port drivers and includes information on the following DVO port drivers:

- CH7009/CH7010 Chronitel* DVO port driver
- CH7017/CH7305 Chronitel* DVO port driver
- FS453/FS454 Focus* DVO port driver

Note: In the following tables, device-specific (non-standard) attributes are highlighted in gray.

C.2.1 Chrontel* CH7009/CH7010 Port Driver TV Attributes

Table 24 lists the TV attributes for the Chrontel* CH7009/CH7010 port drivers.

Table 24. Chrontel* CH7009/CH7010 Port Driver TV Attributes (Sheet 1 of 3)

Registry Entry	Attribute ID	Description	Possible Ranges
BRIGHTNESS	0	Screen brightness	0-100. Default is 50. Recommend 61 for NTSC-M, NTSC-443/433, and PAL-M. Recommend 37 for NTSC-J. Recommend 42 for PAL-B/D/G/H//N formats.
CONTRAST	1	Color contrast. A setting of 0 reduces the contrast, a setting of 1 leaves the image unchanged, and values beyond 1 increase the contrast.	0-7. Default is 3.
FLICKERFILTER	3	TV Flicker Filter. The higher the value, the higher the amount of flicker filtering and text enhancement.	0-1000. Default is 999. Recommend 999 or 1000.
HPOSITION	4	Horizontal Position. Increasing the value moves the image to the right and decreasing the value moves the image to the left.	0-511. Default is 64.
VPOSITION	5	Vertical Position. The value represents the TV line number relative to the VGA vertical sync. Increasing the value moves the image down and decreasing the value moves the image up.	0-511. Default is 0.
TVFORMAT	8	TV Format	1 = NTSC-M 2 = NTSC-M-J 3 = NTSC-433 4= PAL-M 5= PAL-B 6= PAL-G 7= PAL-D 8= PAL-H 9= PAL-I 10= PAL-N 11 = PAL-60 The default is 1, NTSC-M.
ID_DISPLAY	9	Attached display type.	1 = DVI Flat panel 2 = TV Default is 2, TV.

Table 24. Chrontel* CH7009/CH7010 Port Driver TV Attributes (Continued) (Sheet 2 of 3)

Registry Entry	Attribute ID	Description	Possible Ranges
LUMA_FILTER	10	TV Luma Filter used in the scaling and flicker reduction block applied to low and high frequency of the luminance signal according to their specific scaling ratio.	0-3. Default is 3.
CHROMA_FILTER	11	TV Chroma Filter. This value limits the bandwidth of the chroma signal in the CVBS and S-Video output signals. (Refer to Chrontel* CH7009 Data Sheets/Specification for a description of the Video Bandwith Register.)	0 = less bandwidth 1= more bandwidth. Default is 1.
TEXT_FILTER	12	TV Text Filter that controls the text enhancement capability designed in the chip. A value of 0 minimizes the enhancement feature while a value of 7 maximizes it.	0-7. Default is 3.
OUTPUT_TYPES	14	TV Output Types	1 = COMPOSITE 2 = SVIDEO 3 = COMPONENT 4 = CPSTSV (Composite and S-Video) 5 = RGB 6 = RGB+CVBS 7 = SCART1 8 = SCART2 Note: TV Output Types are limited to svideo and composite for the VBIOS.
SCALING_RATIO	19	Overscan scale ratio	1 = 5 to 4 2 = 1 to 1 3 = 7 to 8 (NTSC only) 4 = 5 to 6 5 = 5 to 7 (PAL only) 6 = 5 to 8 7 = 5 to 9 8 = 3 to 4 (NTSC only) 9 = 7 to 10 (NTSC only) 10 = 1 to 2 (NTSC only)

Table 24. Chrontel* CH7009/CH7010 Port Driver TV Attributes (Continued) (Sheet 3 of 3)

Registry Entry	Attribute ID	Description	Possible Ranges
VGA_BYPASS	32769	VGA Bypass. This attribute is associated with the Hardware Configuration attribute.	0 = disable VGA Bypass. 1 = enable VGA Bypass Default is 0, disable VGA Bypass.
DOT_CRAWL	32770	TV Dot Crawl (NTSC only). Dot crawl affects the edges of color and manifests itself as moving dots of color along these edges. A value of 1 freezes dot crawl; 0 allows dot crawl to run freely.	0 = have Dot Crawl run freely 1 = freeze Dot Crawl Default is 0.
HW_CONFIG	32771	Hardware Configuration. This attribute works in conjunction with the VGA Bypass and Display Device attributes.	1-2. Default is 1. If set to 1 and: VGA Bypass = 1 -- Both DVI and VGA are enabled VGA Bypass = 0 -- DVI or TV are enabled If set to 2 and: DVI connection detected -- only DVI is enabled no DVI connection detected and VGA Bypass = 1 -- only VGA Bypass enabled no DVI connection detected and VGA Bypass = 0 -- only DVI enabled.

C.2.2 Chronitel* CH7017/CH7305 Port Driver Attributes

Table 25 shows the attributes for the Chronitel* CH7017 and CH7305 port drivers

Table 25. Chronitel* CH7017/CH7305 Port Driver Attributes (Sheet 1 of 5)

Registry Entry	Attribute ID	Description	Possible Ranges
BRIGHTNESS	0	Screen brightness	0-100. Default is 50. Recommend 61 for NTSC-M, NTSC-443/433, and PAL-M. Recommend 37 for NTSC-J. Recommend 42 for PAL-B/D/G/H/I/N formats.
CONTRAST	1	Color contrast	0-7. Default is 3.
FLICKERFILTER	3	TV Flick Filter	0-1000. Default is 999. Recommend 999 or 1000.
HPOSITION	4	Horizontal Position. This shifts the displayed TV image in horizontally to center the image on the screen. Increasing the value moves the image to the right and decreasing the value moves the image to the left.	0-512. Default is 46. Increment by 1.
VPOSITION	5	Vertical Position. This shifts the displayed TV image in a vertically to center the image. The value represents the TV line number relative to the VGA vertical sync. Increasing the value delays the output of TV vertical sync, thus moving the image down. Decreasing the value moves the image up.	0-512. Default is 6. Increment by 1.
TVFORMAT	8	TV Format	1 = NTSC-M 2 = NTSC-M-J 3 = NTSC-433 4 = PAL-M 5 = PAL-B 6 = PAL-G 7 = PAL-D 8 = PAL-H 9 = PAL-I 10 = PAL-N 11 = PAL-60
ID_DISPLAY	9	Attached display type.	1 = LVDS panel 2 = TV
LUMA_FILTER	10	TV Luma Filter	0-3. Default is 3.

Table 25. Chrontel* CH7017/CH7305 Port Driver Attributes (Continued) (Sheet 2 of 5)

Registry Entry	Attribute ID	Description	Possible Ranges
CHROMAFILTER	11	TV Chroma Filter. This value limits the bandwidth of the chroma signal in the CVBS and S-Video output signals. (Refer to Chrontel* CH7009 Data Sheets/Specification for a description of the Video Bandwidth Register.)	0 = less bandwidth 1 = more bandwidth. Default is 1.
TEXTFILTER	12	TV Text Filter	0-7. Default is 3.
TVOUT_TYPE	14	TV Output Types	1 = COMPOSITE 2 = SVIDEO 3 = COMPONENT 4 = CPSTSV (Composite and S-Video) 5 = RGB 6 = RGB+CVBS 7 = SCART1 8 = SCART2 Note: TV Output Types are limited to svideo and composite for the VBIOS.
FLAT_PANEL_WIDTH	16	Flat panel display width.	0 - 2048. Default 1024. Recommend native panel width.
FLAT_PANEL_HEIGHT	17	Flat panel display height.	0 - 2048. Default 768. Recommend native panel height.
SCALINGRATIO	19	Screen scale ratio	1 = 5 to 4 2 = 1 to 1 3 = 7 to 8 (NTSC only) 4 = 5 to 6 5 = 5 to 7 (PAL only) 6 = 5 to 8 7 = 5 to 9 8 = 3 to 4 (NTSC only) 9 = 7 to 10 (NTSC only) 10 = 1 to 2 (NTSC only)
POWERSEQUENCET1	21	Power Sequencing T1; VDD Active. Defines the power on time T1, the time duration between LVDS_RDY (internal) signal to a valid LVDS clock and data.	1-512. Increment of 1. Default = 255.
POWERSEQUENCET2	22	Power Sequencing T2; Backlight Active. This defines the Backlight Enable Time (T2), the waiting time after a valid LVDS clock and data before enabling the LVDS panel backlight.	2-256; Increment of 2. Default = 127

Table 25. Chrontel* CH7017/CH7305 Port Driver Attributes (Continued) (Sheet 3 of 5)

Registry Entry	Attribute ID	Description	Possible Ranges
POWERSEQUENCET3	23	Power Sequencing T3; Backlight Inactive. This defines the Backlight Disable time (T3), the required time after disabling the backlight before the valid LVDS clock and Data become tri-stated or disabled.	2-256; Increment of 2. Default = 127
POWERSEQUENCET4	24	Power Sequencing T4; VDD Inactive. This defines the Power Off time (T4), the required time prior to power off after the valid LVDS Clock and Data become tri-stated or disabled.	1-512. Increment of 1. Default = 255.
POWERSEQUENCET5	25	Power Sequencing T5; VDD Active to Inactive. This defines the Power Cycle time (T5), the waiting time required prior to enabling power on after power has been off.	1-1600. Increment of 50. Default = 800.
PANELDEPTH	26	Panel Depth defines the picture format, which is either 18 bits or 24 bits.	18 - 24 bits. Increment by 6. Default is 18. Recommend 24 for TV. For flat panels, refer to the panel specification.
SINGLE/DUALCHANNEL	27	Single/Dual Channel defines the chip channel mode.	0 = Single 1 = Dual Default is 0. Recommend 0 for TV. For flat panels, refer to the panel specification.
VGAOUT	32769	Enables or Disables VGA Output, which controls the VGA Bypass associated with the 32771 attribute.	0 = Disable 1 = Enable Default is 0.
DATAPATH	32770	Data Path controls the data path from D1 and D2 input to the internal TV and LVDS blocks.	1 -4 1 = routs D1 to both blocks 2 = routs D1 to LVDS and D2 to TV (DVOB) 3 = routs D1 to TV and D2 to LVDS (DV0C) 4 = routs D2 to both blocks Default is 3.

Table 25. Chrontel* CH7017/CH7305 Port Driver Attributes (Continued) (Sheet 4 of 5)

Registry Entry	Attribute ID	Description	Possible Ranges
TEXTTUNINGCOEFFICIENT	32771	Determines how text is rendered on a display. (Supported only in full-mode.)	0 = Smooth 1 = Normal 2 = Plain 3 = Sharp 4 = Very sharp Default is 1, Normal.
DOT_CRAWL	32772	Dot Crawl affects the edges of color and manifests itself as moving dots of color along these edges. A setting of 1 freezes dot crawl while a setting of 0 allows dot crawl to run freely. (NTSC only)	0 = Have Dot Crawl run freely 1 = Freeze Dot Crawl Default is 0.
LVDS_SPEC	32773	Opens LDI or SPWG. This determines the encoded data format.	0 = Open LDI 1 = Open SPWG Default is 0.
EMICOUPLINGCAP	32776	EMI Coupling CAP (EMI/PLL Setting). This controls the LVDS PLL Capacitor, which allows coupling of any signal to the on chip loop filter capacitor. Note: EMI settings are subject to change upon panel specifications.	1-15. Increment by 1. 1 for 1024x768 panel 9 for 1400x1050 panel 10 for 1600x1200 panel
EMIFDDIVIDER	32777	The EMI FD Divider defines the LVDS PLL Spread Spectrum Frequency Divider Control.	0-2048 If input clock frequency is: 40, enter 51 65, enter 127 108, enter 150 162, enter 290

Table 25. Chrontel* CH7017/CH7305 Port Driver Attributes (Continued) (Sheet 5 of 5)

Registry Entry	Attribute ID	Description	Possible Ranges
EMIFBDIVIDER	32778	The EMI FB Divider defines the LVDS PLL Spread Spectrum Frequency Feedback Divider Control.	0-2048 Increment by 1. Default = 7.
EMILOOPFILTERRESISTOR	32779	The EMI Loop Filter Resistor controls the LVDS PLL Loop Filter Resistor.	Range from 0-7 0 = 1800 Ohms 1 = 2600 Ohms 2 = 1000 Ohms 3 = 2300 Ohms 4 = 21,800 Ohms 5 = 42,600 Ohms 6 = 11,000 Ohms 7 = 73,200 Ohms Default is 2. Recommend 3 if IEGD 3.3 driver is installed. Recommend 2 if IEGD 5.0 driver is installed.
EMIREDUCTIONRESISTOR	32780	The EMI Reduction Resistor determines the Spread Spectrum Oscillator Frequency setting or the amplitude of an external frequency source.	Range from 0-7 0 = 0uA 1 = 10uA 2 = 20uA 3 = 30uA 4 = 40uA 5 = 50uA 6 = 60uA 7 = 70uA Default is 3. Recommend 4 if IEGD 3.3 driver is installed. Recommend 3 if IEGD 5.0 driver is installed.

C.2.3 Chrontel* CH7307 Port Driver Attributes

Table 25 shows the attributes for the Chrontel* CH7307 port driver.

Table 26. Chrontel* CH7017/CH7307 Port Driver Attributes

Registry Entry	Attribute ID	Description	Possible Ranges
FLAT_PANEL_WIDTH	16	Flat panel display width.	0 - 2048. Default 1024. Recommend native panel width.
FLAT_PANEL_HEIGHT	17	Flat panel display height.	0 - 2048. Default 768. Recommend native panel height.
Spread Spectrum Clocking	39	Spectrum clocking	0-15 Default = 0 Step = 1

C.2.4 Chronitel* CH7308 Port Driver Attributes

Table 25 shows the attributes for the Chronitel* CH7308 port driver.

Table 27. Chronitel* CH7308 Port Driver Attributes

Registry Entry	Attribute ID	Description	Possible Ranges
FLAT_PANEL_WIDTH	16	Flat panel display width.	0 - 2048. Default = 1024. Recommend native panel width.
FLAT_PANEL_HEIGHT	17	Flat panel display height.	0 - 2048. Default = 768. Recommend native panel height.
FLAT_PANEL_POWER_T1	21	VDD Active - DVO CLK/ DATA active.	0 - 1023 Default = 25 Step = 1
FLAT_PANEL_POWER_T2	22	DVO active - Backlight enable.	0 - 1023 Default = 250 Step = 1
FLAT_PANEL_POWER_T3	23	DVO active - Backlight disable.	0 - 1023 Default = 250 Step = 1
FLAT_PANEL_POWER_T4	24	DVO inactive - VDD inactive.	0 - 1023 Default = 25 Step = 1
FLAT_PANEL_POWER_T5	25	VDD inactive - VDD active	0 - 1023 Default = 400 Step = 1
LVDS Color Depth	26		1 = 18 bits 2 = 24 bits Default = 1
DUAL_CHANNEL	27	Dual-channel pane	Default - 0
Spread Spectrum Clocking	39	Spectrum Clocking	0-15 Default = 0 Step = 1
Dither	41	Dither setting	Default = 0
HSync Panel Protection	42	Horizontal sync panel protection	Default = 0
VSyn Panel Protection	43	Vertical sync panel protection	Default = 0
Pixel Clock Protection	44	Pixel clock protection	Default = 0
LVDS Panel Connector	45	LVDS panel connector.	1 = SPWG 2 = OpenLDI Default = 1

C.2.5 Focus* FS453/FS454 Port Driver TV Attributes

Table 28 lists the TV attributes for the Focus* FS453 and FS454 port drivers.

Table 28. Focus* FS453/FS454 Port Driver TV Attributes (Sheet 1 of 2)

Registry Entry	Attribute ID	Description	Possible Ranges
BRIGHTNESS	0	Screen Brightness	0-100
CONTRAST	1	Color Contrast	0-100
FLICKER	3	Flicker Filter	0-1000, default is 1000
HPOSITION	4	Horizontal Position	0-100
VPOSITION	5	Vertical Position	0-100
HSCALE	6	Horizontal Scale	0-1000, increment by 10.
VSCALE	7	Vertical Scale	0-1000, increment by 10.
TVFORMAT	8	TV Format	1 = NTSC 2 = NTSC-EIAJ 3 = PAL 4 = PAL-D 5 = PAL-G 6 = PAL-H 7 = PAL-I 8 = PAL-M 9 = PAL-N 10 = PAL-NC 11 = PAL-60 12 = 480P 13 = 720P
LUMAFILTER	10	Luma Filter	0 = Disabled 1 = Enabled
CHROMAFILTER	11	Chroma Filter	0 = Disabled 1 = Enabled
MACROVISION	13	Video Copy Protection.	
TVOUTTYPE	14	TV Output Format	1 = CVBS 2 = CVBS+Y/C 3 = Y/C 4 = YPrPb 5 = SCART-RGB 6 = VGA-DAC Note: TV Output Types are limited to svideo and composite for the VBIOS.
SATURATION	15	Color Saturation	0-100. Default is 50.
REVERSE DVO DATA	30	Reverses the data order coming out of the DVO port.	0 = Normal DVO data order 1 = Reverse the DVO RGB data order. Default is 0.

Table 28. Focus* FS453/FS454 Port Driver TV Attributes (Continued) (Sheet 2 of 2)

Registry Entry	Attribute ID	Description	Possible Ranges
SHARPNESS	31	Image Sharpness	0-1000. Default is 800.
HWCONFIG	32	Hardware Configuration. This attribute allows you to select a hardware configuration for the FS453/FS454 port drivers.	0 = Standard configuration 1 = FS454 ADD card 2 = ADD card (Japan J connector) 3 = Standard configuration (invert clock) Default is 3 for the FS454.
HORZFILTER	33	Horizontal Filter	0-100. Default is 50.

C.2.6 National Semiconductor* NS387R Port Driver LVDS Attributes

Table 29 lists the LVDS attributes for the National Semiconductor* NS387R port driver.

Table 29. National Semiconductor* NS387R Port Driver LVDS Attributes

Registry Entry	Attribute ID	Description	Possible Ranges
FP WIDTH	16	Flat Panel Width. Note: The NS387R does not support scaling. In this case, FP Width and Height are used to limit the number of modes exported to the panel. Only matching modes with FP Width and Height are exported.	0 - 2048 Default = 0
FP HEIGHT	17	Flat Panel Height	0 - 2048 Default = 0
GANG MODE	28	Gang display or not.	0 = Off 1 = On Default = 0
GANG MODE EVEN ODD	29	Gang display even or odd.	0 = Even 1 = Odd Default = 0
REVERSE DVO DATA	30	Reverses the data order coming out of the DVO port.	0 = Normal DVO data order 1 = Reverse the DVO RGB data order. Default is 0.

C.2.7 Silicon Image* Sil164 Port Driver DVI Attributes

Table 30 lists the DVI attributes for the Silicon Image* Sii164 port driver.

Table 30. Silicon Image* Sil164 Port Driver DVI Attributes

Registry Entry	Attribute ID	Description	Possible Ranges
FP WIDTH	16	Flat Panel Width Note: The Sii164 does not support scaling. In this case, FP Width and Height are used to limit the number of modes exported to the panel. Only matching modes with FP Width and Height are exported.	0 - 2048 Default = 0
FP HEIGHT	17	Flat Panel Height	0 - 2048 Default = 0
EDGE SELECT	32769	Allows you to invert the input clock signal.	0 = Input data is falling edge latched 1 = Input data is rising edge latched Default = 0

C.2.8 Silicon Image* Sil 1362/Sil 1364Port Driver DVI Attributes

Table 30 lists the DVI attributes for the Silicon Image* Sil 1362/1364 port drivers.

Table 31. Silicon Image* Sil 1362/Sil 1364 Port Driver DVI Attributes

Registry Entry	Attribute ID	Description	Possible Ranges
FP WIDTH	16	Flat Panel Width	0 - 2048 Default = 0 Step = 1
FP HEIGHT	17	Flat Panel Height	0 - 2048 Default = 0 Step = 1

C.3 Chipset and Port Driver-Specific Installation Information

Table 32. Default DVO Search Order

Chipset	Default DVO Search Order
815/815E	DVO
845GV	DVOC, DVOB
85xGME	DVOC, DVOB, LVDS
852GM	DVOC, DVOB, LVDS
915GV	sDVOB, sDVOC
915GM	sDVOB, sDVOC, LVDS
945G	sDVOB, sDVOC
945GM	sDVOB, sDVOC, LVDS

Table 33. Default GPIO Pin Pair Assignments

Chipset	Default GPIO Pin Pair for EDID			
	DVO/A	DVOB	DVOC	LVDS
815E	0	N/A	N/A	N/A
845GV	N/A	3	3	N/A
85xGME	2	3	3	2
852GM	2	3	3	2
915GV	N/A	4	4	N/A
915GM	N/A	4	4	2
945G	N/A	4	4	N/A
945GM	N/A	4	4	2

Table 34. Default IC2 Device Address Byte Assignment (Sheet 1 of 2)

Port Driver	Default Device Address Bytes (DAB)
CH7009	0xEC
CH7307	0x70 (for first sDVO device) 0x72 (for second sDVO device)
CH7308	0x70 (for first sDVO device) 0x72 (for second sDVO device)
FS454	0x94
NS2501	0x70
NS387R	0x70
Sil164	0x70

Table 34. Default IC2 Device Address Byte Assignment (Continued) (Sheet 2 of 2)

Sil1362	0x70 (for first sDVO device) 0x72 (for second sDVO device)
Sil1364	0x70 (for first sDVO device) 0x72 (for second sDVO device)
TH164	0x70

This page intentionally left blank.

Intel 5F Extended Interface Functions D

The BIOS provides a set of proprietary function calls to control operation of the extended features. These function calls all use AH = 5Fh in their designed interface for easy identification as a proprietary function

These functions are designed to maintain maximum compatibility with the Desktop and Mobile Video BIOS. As such many of the definitions behave identically. When the behavior of the Embedded Video BIOS is not identical to the Desktop and Mobile Video BIOS it is noted.

In addition to these 5F functions, the Video BIOS also supports all 4F functions defined by the *VESA BIOS Extension (VBE) Core Functions Standard, Version 3.0* with the exception of the 0A function (Return VBE Protected Mode Interface). All other functions, from 00 through 09 and 0B are supported by the Video BIOS. Click on the following link to view the VBE 3.0 Core Functions Standard document.

<http://www.vesa.org/Public/VBE/vbe3.pdf>

Table 35 provides a summary of the IEGD supported Intel 5F functions.

Table 35. Summary of Intel 5F Extended Interface Functions

Function	Function Name	Description
BIOS Extended Interface Functions		
5F01h	Get Video BIOS Information	Gets VBIOS Build Information.
5F05h	Refresh Rate	Sets a new vertical refresh rate for a given mode and returns the current vertical refresh rate
5F10h	Get Display Memory Information	Returns information about the linear memory.
5F1Ch	BIOS Pipe Access	Sets the BIOS pipe access and returns the BIOS pipe access status.
5F29h	Get Mode Information	Returns information on the requested mode.
5F61h	Local Flat Panel Support Function	Supports local flat panel features.
Hooks for the System BIOS		
5F31h	POST Completion Notification Hook	Signals the completion of video POST (Power On Self Test)
5F33h	Hook After Mode Set	Allows System BIOS to intercept Video BIOS at the end of a mode set.
5F35h	Boot Display Device Hook	Allows System BIOS to override video display default setting.
5F36h	Boot TV Format Hook	Allows System BIOS to boot TV in selected TV format state.
5F38h	Hook Before Set Mode	Allows System BIOS to intercept Video BIOS before setting the mode.
5F40h	Config ID Hook	Allows System BIOS to supply a configuration ID that is passed to the driver.

This page intentionally left blank.

Index

Numerics

5F functions
table of 183

A

Acronyms and Terminology 14
Advanced EDID Configuration 50
AGP Aperture Size 28
API support 23
Architecture
 IEGD driver 20
 IEGD suite overview 19
-asm option
 pcf2iegd utility 29
Assembler
 MASM* 72
Assignment syntax
 PCF file 31
Attributes
 CH7009/CH7010 port driver, table of 165
 CH7017/CH7305 port driver, table of 168
 CH7308 port driver, table of 173, 174
 FS453/FS454 port driver, table of 175, 177, 178
 Sil 1362/Sil 1364 port driver, table of 179
 standard port driver, table of 163

B

Backward compatibility 85, 117
-bin option
 pcf2iegd utility 31
Blend filtering
 Windows CE* OS 115
Block heading hierarchy
 PCF file 32
Block syntax
 in PCF file 31
 (See also PCF file.)
BOOT display attachment selection category
 def_eg.txt configuration file 81
 (See also Legacy Video BIOS.)
Build selection category
 def_eg.txt configuration file 73
 (See also Legacy Video BIOS.)
Building the Video BIOS 59
 prerequisites 60
 procedure 63
Built-in display modes 50

C

Catalogue feature file (igd.cec)
 Windows CE* OS 116
-cfg option
 pcf2iegd utility 30
CH7009/CH7010 port driver attributes, table of 165
CH7017/CH7305 port driver attributes, table of 168
CH7308 port driver attributes, table of 173, 174
Chipsets
 default DVO search order 180
 default GPIO pin pair assignments 180
 IEGD support 22
 Intel® 855GME dual display parameter settings 86
COMMON_TO_PORT parameter 61
Compilers
 Watcom* C/C++ compiler 60, 72
Component Designer 92
Configuration
 advanced EDID 50
 dual panel configuration in Windows* OS 86
 Linux OS 125
 Linux* OS
 driver options, table of 136
 PCF file
 general format rules 31
 platform 27
 universal INF 85
 VBIOS and Driver 28
 Windows CE* OS 103
 Windows* OS 89
 XFree86 and Xorg 136
configuration file

D

def_eg.txt configuration file 73
 BOOT display attachment selection category 81
 build selection category 73
 general selection category 75
 PORT_CONFIG selection category 76
 PORT_DEVICE selection category 76
 version selection category 73
Default display mode
 changing 92
Default mode parameters 62
default_refresh parameters 63
default_vga_height parameter 63
Detectable displays 49
Device Address Byte
 default assignments 180
Directory structure
 Video BIOS 61

- Display Configurations
 - types 21
- Display configurations
 - supported configurations 22
- Display detect operation 47
- Display Detection and Initialization 47
- Display modes
 - built-in 50
 - EDID-DTDs 50
 - user-specified DTDs 50
- DisplayDetect key
 - in ssigd.inf file 47
- Displays
 - configurations 21
 - detectable 49
 - supported types 21
- Downloading IEGD Driver and Video BIOS 16
- DTD settings
 - flags parameter in PCF file 40
 - in PCF file 39
- Dual panel configuration
 - Intel® 855GME chipset example 86
 - Windows* OS 86
- DVO devices
 - description 24
- DVO ports
 - default search order 180
- Dynamic port driver loading
 - Windows CE* OS 116
- Dynamic port drivers
 - See Port drivers.

E

- EDID-DTDs 50
- EDID-Less configuration 23
 - Linux* OS 140, 185
- EDID-Less file format
 - Linux* OS 140
- EDID-Less panel type detection
 - description 23
- Error messages
 - pcf2iegd
 - Error, no input file specified 31
- Examples
 - device configuration for Linux* OS 138
 - Intel® 855GME dual display parameter settings 86
- External PCI Graphics Adaptor 51

F

- Files 73
 - .sld file 91
 - EDID-.bin 140
 - igd.cec 116
 - INF
 - backward compatibility 85
 - sample 155
 - Linux configuration file 134

- PCF file 31
- PCF file sample 42
- ssigd.inf
 - DisplayDetect key 47
- video.reg 103
 - file backward compatibility 117
 - sample 118
- XF86.Conf file
 - DisplayDetect option 47
- XF86Conf sample 132
- Xorg.conf file
 - DisplayDetect option 47
- Flags parameter
 - in PCF file 40
- Framebuffer
 - Windows CE* OS 109
- FS453/FS454 port driver attributes, table of 175, 177, 178

G

- General selection category
 - def_eg.txt configuration file 75
 - (See also Legacy Video BIOS.)
- GMCH
 - as secondary graphics device 51
 - PCI device enabling 27
- GPIO pin pair
 - default assignments 180
- Graphics and Memory Controller Hub (GMCH) See GMCH
- Graphics Mode Select (GMS) 27
- Graphics port initialization
 - Linux* OS 139

H

- h option
 - pcf2iegd utility 29

I

- IEGD driver
 - architecture 20
 - compatibility with Video BIOS 63
 - configuration 28
 - configuration procedure 17
 - definition 11
 - detectable displays 49
 - display configuration 21
 - display detection and initialization 47
 - display mode configuration for Windows CE* OS 111
 - downloading 16
 - DVO devices 24
 - EDID-less configuration 23
 - general configuration section for Windows CE* OS 110
 - installing on Windows* OS 92
 - new features at Version 4.0 13
 - options for Linux* OS
 - table of 136
 - OS and API support 23

- PCF file 31
- pcf2iegd utility 29
- platform configuration 27
- rotation support 24
- runtime configuration GUI 95
- supported chipsets 22
- supported display types 21
- uninstalling from Windows* OS 94
- IEGDGUI runtime configuration utility
 - Linux* OS 143
 - setup 143
 - using 144
- igd.cec catalogue feature file
 - Windows CE* OS 116
- INF file
 - backward compatibility 85
 - sample 155
- inf option
 - pcf2iegd utility 30
- Installation
 - IEGD driver on Linux* OS 126
 - IEGD driver on Windows CE* OS 102
 - IEGD driver on Windows* OS 92
 - Linux OS 125
- Intel 5F Extended Interface Functions See 5F functions
- Intel® 855GME chipset
 - dual display parameter settings example 86

K

- Keys
 - PcfVersion 86, 117

L

- Legacy Video BIOS 69
 - (See also Video BIOS.)
 - compatibility with IEGD driver 82
 - configuration tips 82
 - deg_eg.txt configuration file 73
 - model 70
 - panel detection 71
 - User Build System (UBS) 72
 - VBIOS customization tool (VCT) 81
 - VESA modes 84
 - video modes 82
- Linux OS
 - configuration file 134
 - installation and configuration 125
 - XFree86 and Xorg configuration 136
- Linux* OS
 - EDID-.bin file 140
 - EDID-Less configuration 140, 185
 - EDID-Less file format 140
 - example device configuration 138
 - graphics port initialization 139
 - IEGD driver options
 - table of 136
 - IEGDGUI setup 143

- installation prerequisites 125
- installation procedure 126
- modelines 140
- runtime configuration GUI 143
- using IEGDGUI 144
- XF86.Conf file
 - DisplayDetect option 47
- Xinerama 139
- Xorg.conf file
 - DisplayDetect option 47

M

- MASM* assembler 72
- Modelines 140
- Modes
 - VESA and VGA 64

N

- NOTEARING support
 - Windows CE* OS 116

O

- OEM vendor string parameters 62
- Options
 - driver options for Linux* OS 136
- OS support 23

P

- Panel Detection
 - legacy Video BIOS 71
- Parameter Configuration Format (PCF) file 31
- PCF file
 - assignment syntax 31
 - block heading hierarchy 32
 - block headings hierarchy 31
 - block syntax 31
 - COMMON_TO_PORT parameter 61
 - configuring for the Video BIOS 61
 - default mode parameters 62
 - default_refresh parameters 63
 - default_vga_height parameter 63
 - dtd parameter 39
 - error message 31
 - general format rules 31
 - OEM vendor string parameters 62
 - parameters
 - table of 33
 - post_display_msg parameter 62
 - sample 42
- pcf2iegd utility
 - asm option 29
 - bin option 31
 - cfg option 30
 - error message 31

- h option 29
 - inf option 30
 - usage 29
 - VBIOs and driver configuration 29
 - ver option 29
 - x option 30
 - PcfVersion key 86, 117
 - PCI device enabling 27
 - PCI Graphics Adaptor 51
 - Per port customization
 - Windows CE* OS
 - attribute initialization 114
 - custom DTD mode timings 113
 - custom DVO GPIO Pin Settings 114
 - custom flat panel controls 114
 - Pin pair definitions
 - table of 41
 - Platform
 - per port customizations
 - Windows CE* OS 112
 - Platform configuration 27
 - Port drivers
 - CH7009/CH7010 attributes, table of 165
 - CH7017/CH7305 attributes, table of 168
 - CH7307 attributes, table of 173
 - CH7308 attributes, table of 174
 - default Device Address Byte assignments 180
 - FS453/FS454 attributes, table of 175, 177, 178
 - installation options on Windows CE* OS 107
 - Sil 1362/Sil 1364 attributes, table of 179
 - standard attributes, table of 163
 - Windows* OS configuration 89
 - PORT_CONFIG selection category
 - def_eg.txt configuration file 76
 - (See also Legacy Video BIOS.)
 - PORT_DEVICE selection category
 - def_eg.txt configuration file 76
 - (See also Legacy Video BIOS.)
 - post_display_msg parameter 62
 - Prerequisites
 - for building Video BIOS 60
 - Windows CE* OS installation 101
- R**
- Registry keys
 - Windows CE* OS 104
 - Reserved memory
 - Windows CE* OS 108
 - Rotation 24
 - Runtime configuration GUI
 - Linux* OS 143
 - Windows* OS 95
- S**
- Sample files
 - INF 155
 - video.reg 118
 - XF86Conf 132
 - Sil 1362/Sil 1364 port driver attributes, table of 179
 - sld file
 - for Windows XP Embedded* OS 91
 - ssigd.inf file
 - DisplayDetect key 47
 - Stolen memory
 - Graphics Mode Select settings 27
 - System BIOS
 - GMS settings 27
 - interface with legacy Video BIOS 82
 - settings 27
 - System requirements
 - Video BIOS 59
 - System to video stretch blit
 - Windows CE* OS 115
- T**
- Terminology 14
 - Text anti-aliasing
 - Windows CE* OS 116
- U**
- Uninstalling IEGD driver
 - Windows* OS 94
 - Universal INF configuration 85
 - User Build System (UBS)
 - building the video BIOS 59
 - legacy Video BIOS 72
 - User-specified DTDs 50
 - Utilities
 - IEGDGUI
 - Linux* OS 143
- V**
- VBIOs Customization Tool (VCT)
 - legacy Video BIOS 81
 - ver option
 - pcf2iegd utility 29
 - Version selection category
 - def_eg.txt configuration file 73
 - (See also Legacy Video BIOS.)
 - Vertical Extended Display Mode 112
 - VESA modes
 - legacy Video BIOS support 84
 - table of 64
 - Video BIOS support 65
 - VGA modes
 - legacy Video BIOS 82
 - table of 64
 - Video BIOS 59
 - (See also Legacy Video BIOS.)
 - building 63
 - COMMON_TO_PORT parameter 61
 - configuration 28
 - configuration procedure 17

- configuring the PCF file. 61
- default mode parameters 62
- default_refresh parameters 63
- default_vga_height parameter 63
- directory structure 61
- downloading 16
- driver compatibility 63
- EDID-less configuration 23
- legacy 69
- OEM vendor string parameters 62
- pcf file 31
- pcf2iegd utility 29
- platform configuration 27
- post_display_msg parameter 62
- system requirements 59
- VESA mode support 65
- zip file 60
- Video surface allocation rule
 - Windows CE* OS 110
- video.reg file 103
 - backward compatibility 117
 - sample 118

W

- Watcom* C/C++ compiler 60, 72
- Windows XP Embedded* OS
 - Component Designer 92
- Windows CE* OS
 - attribute initialization 114
 - backward compatibility 117
 - blend filtering 115
 - catalogue feature file (igd.cec) 116
 - configuration 103
 - custom DTD mode timings 113
 - custom DVO GPIO pin settings 114
 - custom flat panel controls 114
 - determining graphics memory size 109
 - display mode configuration 111
 - dynamic port driver loading 116
 - framebuffer and video surface size 109
 - general configuration selection 110
 - installation 102
 - installation prerequisites 101

- NOTEARING support 116
- PcfVersion key 117
- per port platform customization 112
- port driver installation options 107
- registry keys, table of 104
- reserved memory model 108
- sample video.reg file 118
- system to video stretch blit 115
- text anti-aliasing 116
- Vertical Extended Display Mode 112
- video surface allocation rule 110
- video.reg file 103
- Windows XP Embedded* OS
 - creating an .sld file 91
- Windows* OS
 - backward compatibility 85
 - configuring and installing 85
 - dual panel configuration 86
 - installing IEGD driver 92
 - PcfVersion key 86
 - runtime configuration GUI 95
 - uninstalling the IEGD driver 94
 - universal INF configuration 85

X

- x option
 - pcf2iegd utility 30
- XF86.Conf file
 - DisplayDetect option 47
- XF86Conf file 132
- XFree86
 - configuration 136
- Xinerama
 - Linux* OS 139
- Xorg
 - configuration 136
- Xorg.conf file
 - DisplayDetect option 47

