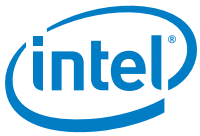

Intel® Digital Set Top Box Display Driver

User's Guide for Microsoft* Windows* CE 5.0

January 2006

Driver 1-0-0-0094

Legal Statements

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 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.

This manual 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.

This manual 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 document 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.

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino logo, Chips, Core Inside, Dialogic, EtherExpress, ETOX, FlashFile, i386, i486, i960, iCOMP, InstantIP, Intel, Intel logo, Intel386, Intel486, Intel740, IntelDX2, IntelDX4, IntelSX2, Intel Core, Intel Inside, Intel Inside logo, Intel. Leap ahead., Intel. Leap ahead. logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Viiv, Intel XScale, IPLink, Itanium, Itanium Inside, MCS, MMX, MMX logo, Optimizer logo, OverDrive, Paragon, PDCharm, Pentium, Pentium II Xeon, Pentium III Xeon, Performance at Your Command, Pentium Inside, skoool, Sound Mark, The Computer Inside., The Journey Inside, VTune, Xeon, Xeon Inside 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.

Contents

Revision History

Date	Driver	Description
May 2005	1-0-0-0064	This is the first version of the new guide. The content and layout are all new.
18 May 2005		Updated after technical review.
July 2005		Added hardware-accelerated bob support for overlay surfaces with interleaved data using DirectDraw. Added new samples (BobTest, CapsTest, PerPixelAlphaBltTest, VBITest).
December 2005	1-0-0-0092	Updated escape interface for HDMI Hotplug and HDMI/sil9030 over scan flag. Added new sample (ClipBltTest). Added overlay formats that support progressive and interleaved data, and noted YUV 4:2:0 planar limitation for 830 and 854.
January 2006	1-0-0-0094	Removed the content for 815. Added details of DAC configuration for different TV out type for CX25899 in HH2 TV card.

This page intentionally left blank

1 Introduction

This document describes the Intel® Digital Set Top Box Display Driver for Microsoft* Windows* CE 5.0. It provides a summary of display driver features, installation notes, and configuration information. It 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, and system control application developers.

1.1 Definitions

Table 1. Definitions

Term	Definition
API	Application Programming Interface
BLT	Bit block transfer
BPP	Bits per pixel
DDGPE	DirectDraw* Graphics Primitive Engine
Direct3D*	An API for manipulating and displaying three-dimensional objects. Developed by Microsoft, Direct3D provides programmers with a way to develop 3-D programs that can utilize whatever graphics acceleration device is installed in the machine. Virtually all 3-D accelerator cards for PCs support Direct3D. Another 3-D standard offering similar functionality is OpenGL.
GDI	Graphical Device Interface
GMCH	Graphics Memory Controller Hub
GWES	Graphics, Windowing, and Events Subsystem
VBI	Vertical Blanking Interval

2 Feature Summary

2.1 GMCHs

The Intel® Digital Set Top Box Display Driver supports the following Graphic Memory Controller Hubs:

- Intel® 82830M GMCH
- Intel® 82854 GMCH

2.1.1 Base Features

2.1.1.1 Common Base Features

The Intel® Digital Set Top Box Display Driver includes support for:

- DirectDraw 2D APIs, including: DdCreateSurface, DdDestroySurface, DdLock, DdBlt and DdAlphaBlt, and DdUnlock.
- DirectDraw Video APIs, including: DdCreateSurface, DdDestroySurface, DdSetColorKey, DdLock, DdUpdateOverlay, DdSetOverlayPosition, DdFlip, and DdUnlock.
- DDGPE APIs, which extend GPE classes to support DirectDraw.
- GDI APIs, including hardware accelerated BLT support: color fill, source copy, stretch, and system to video memory.
- Power Management, including support for Full On and Suspend (OS power states D0 and D4).
- ExtEscape interface – installed in the Extras\src\inc folder by the display driver MSI.
- Dynamic memory configuration – for when the amount of video memory is not a significant concern. This model has no limit on the amount of video memory that can be allocated in the system, besides the amount of available memory.

2.1.2 830 and 854 Base Features

The Intel® Digital Set Top Box Display Driver for the 830 and 854 includes support for:

- Region alpha blending using the DirectDraw method AlphaBlt — where a single alpha value is used for the entire rectangle.
- Per-pixel alpha blending using the DirectDraw method AlphaBlt — where a separate alpha value is used for each pixel.
- Plane based alpha blending, which uses the GMCH to blend the graphics plane with the overlay plane. See the AlphaBlendMode registry option in 4.1.1 Display Options for more information.
- Blend options, including bilinear and anisotropic.
- Direct 3D Mobile
- Hardware anti-aliased text
- Sample applications – installed in the Extras folder by the display driver MSI. Samples are CETK DLLs. An executable file called TestShell is also provided that can execute the CETK DLLs without using CETK.
- Hardware-accelerated bob for overlay surfaces with interleaved data using DirectDraw (DDCAPS2_CANBOBINTERLEAVED and DDCAPS2_CANFLIPODDEVEN caps are set). Example usage is provided in the BobTest sample. Note: IDirectDrawSurface5::UpdateOverlay DDOVER_BOBHARDWARE flag is not supported since DirectDraw Video Port Extensions are not supported.

2.2 Surface Support

The display driver includes support for:

- Single, double and triple frame buffer options
- Planar 4:2:0 YV12 overlay format with progressive data
- Planar 4:2:0 I420 overlay format with progressive data
- Planar 4:2:0 CMWH overlay format with progressive data (for HWMC)
- Packed 4:2:2 YUY2 overlay format with interleaved and progressive data
- Packed 4:2:2 UYVY overlay format with interleaved and progressive data
- Packed 4:2:2 YVYU overlay format with interleaved and progressive data
- Packed 4:2:2 VYUY overlay format with interleaved and progressive data
- Packed xRGB8888 overlay format with interleaved and progressive data
- Single, double, and triple video overlay options

Note: For planar YUV 4:2:0 overlay formats, interleaved data is displayed with the temporal chroma artifact commonly referred to as the Chroma Upsampling Error (CUE).

2.3 Display Configuration

The display driver includes support for the following resolutions. (In the following list 60Hz is used as a short-hand for 59.94Hz.)

For use with NTSC:

- 640x480 @ 60Hz – scaled to 720x480
- 720x480 @ 60Hz
- 800x600 @ 60Hz – scaled to 720x480
- 1024x768 @ 60Hz – scaled to 720x480

For use with PAL and SECAM:

- 640x480 @ 50Hz – scaled to 720x576
- 720x576 @ 50Hz
- 800x600 @ 50Hz – scaled to 720x576
- 1024x768 @ 50Hz – scaled to 720x576

For use with Extended Definition 480p:

- 720x480 @ 60Hz

For use with Extended Definition 576p:

- 720x576 @ 50Hz

For use with High Definition – 720p

- 1280x720 @ 60Hz

For use with High Definition 1080i

- 1920x1080 @ 60Hz

The display driver includes support for the following bit depths:

- 16 bits per pixel
- 32 bits per pixel (Intel 82830M GMCH and Intel 82854 GMCH only)

The display driver includes support for the following display types:

- CRT and digital flat panels – via the analog(VGA) port
- TV – with the use of external TV encoders via a DVO port

The display driver includes support for the following display modes:

- Analog(VGA) – output to the analog(VGA) port
- DVO – output to the DVO port
- Twin – output to both the analog(VGA) port and DVO port using DVO timings

2.3.1 Display Configuration Notes

Note: The TV Format and resolution cannot be changed at run-time. Any changes must be made in the video.reg registry file.

Note: Expect TVs to crop the desktop as TVs normally crop their output. Monitors that are normally attached to PCs are not likely to crop the desktop. Monitors accepting TV input may crop the desktop. For that reason, avoid placing user interface elements near the edge of the display.

Note: Some monitors that accept DVI input expect 1280x768 or 1024x768 and may not work with 1280x720 (which is the resolution for 720p).

2.4 VBI Support

The display driver includes support for the following VBI features:

- Closed Captioning with NTSC (Line 21, field 1 & 2)
- Copy Generation Management System on NTSC (Line 20, field 1; Line 21, field 2)
- Wide Screen Signaling on NTSC (Line 20, field 2; Line 20, field 1)

2.5 TV Encoders

The display driver has been tested with the following TV encoders connected to the DVO port.

TV Encoder	Tested on the 830	Tested on the 854
Focus FS454	Yes	Yes
Conexant* CX25873	Yes	No
Conexant CX25892	No	Yes
Silicon Image SiI9030	No	Yes

2.6 Direct3D Mobile Support

The display driver includes support for the following subset of the Direct3D Mobile APIs on the 830 and 854.

- Returning capabilities including GetDeviceCaps
- Index buffers
- Vertex buffers
- Textures
- Methods for drawing including DrawPrimitive and DrawIndexedPrimitive
- Render states
- 4 texture blending stages
- Lighting

3 Installation Notes

3.1 Display Driver MSI

The display driver MSI installs driver files to the $(_WINCEROOT)\IntelCEG\Drivers\Display directory.

3.1.1 830 MSI

The 830 MSI installs the display driver DLL, the display driver D3D Mobile DLL, video.reg, the Conexant cx873 DLL, and the Focus fs454 DLL to the I830 subdirectory. It installs display driver include files to the I830\extras\src\inc subdirectory. It also installs sample application source to I830\extras\src\common and I830\extras\src\samples and sample application binaries to I830\extras\bin.

Note: The Conexant cx873 DLL supports both the CX25873, CX25892, and CX25899 TV encoders.

3.1.2 854 MSI

The 854 MSI installs the display driver DLL, the display driver D3D Mobile DLL, video.reg, the Conexant cx873 DLL, the Focus fs454 DLL, and the Silicon Image SiI9030 DLL to the I854 subdirectory. It installs display driver include files to the I854\extras\src\inc subdirectory. It also installs sample application source to I854\extras\src\common and I854\extras\src\samples and sample application binaries to I854\extras\bin

Note: The Conexant cx873 DLL supports both the CX25873, CX25892, and CX25899 TV encoders.

3.1.3 Installing the MSI

Before installing the display driver MSI, remove any older versions of the display driver:

1. Exit Platform Builder, if necessary.
2. Click **Start > Control Panel > Add or Remove Programs** to remove the previous drivers.
3. Delete the driver files found in $(_WINCEROOT)\INTELCEG\drivers.

To install the new Display Driver MSI, double-click the .msi file.

3.2 Driver Files

The display driver and TV encoder DLLs must be included in the Windows CE image. To do this, make sure either Platform.bib or Project.bib includes statements for the display driver DLL and the TV encoder DLL. A Direct3D Mobile implementation module can be included optionally

For example, to configure the display driver and the Focus FS454 DLL:

```
#define   INTELCEG_DRIVERS_DIR $(_WINCEROOT)\INTELCEG\drivers
ddi_iceg.dll  $(INTELCEG_DRIVERS_DIR)\display\i854\ddi_iceg.dll     NK  SH
fs454.dll     $(INTELCEG_DRIVERS_DIR)\display\i854\fs454.dll        NK  SH
```

To configure the Direct 3D module add:

```
gdld3dm.dll   $(INTELCEG_DRIVERS_DIR)\display\i854\gdld3dm.dll      NK  SH
```

3.3 Registry Options

The MSI installs a configuration file that contains registry options. If you wish to keep this file separated, please make sure this file is included in either Platform.reg or Project.reg. For example:

```
#define   INTELCEG_DRIVERS_DIR $(_WINCEROOT)\INTELCEG\drivers
#include  "$(INTELCEG_DRIVERS_DIR)\Display\I854\video.reg"
```

It is also possible to copy the contents of this fileinto the end of Platform.reg or Project.reg.

The registry settings in video.reg must be modified to configure the display driver and TV Encoder DLL to use on the target platform.

4 Configuration Information

The display driver and TV Encoder DLL configuration use registry settings found in video.reg. The following sections describe the available options. This section is a companion to, and not a replacement for, video.reg. See video.reg for acceptable values and default values for each of the options described below.

4.1 Display Driver Options

4.1.1 Display Options

The display options configure the display resolution and mode.

[HKEY_LOCAL_MACHINE\System\GDI\Drivers]

Display
This key is set to ddi_iceg.dll to enable GWES to load our display driver.

[HKEY_LOCAL_MACHINE\Drivers\Display\Intel]

Width <DWORD>
Requested width of the frame buffer – in pixels.

Height <DWORD>
Requested height of the frame buffer – in active lines.

Depth <DWORD>
Requested pixel depth of the frame buffer – in bits per pixel.

Refresh <DWORD>
Requested rate at which the frame buffer is sent out from the GMCH to the output port.

DVOScanMode <DWORD>
Bit Field that specifies which timing table flags to preset when matching compatible timing settings in port-driver timing tables. Available flags are (Interlaced, Pixel Double, and Line Double). This value should be set to 0x03 (that is, Interlaced and Pixel Doubled output) when specifying 480i/576i HDMI only mode. It should be set to 0x01 (Interlaced) when setting 1080i mode for any TV encoder.

DigitalPort <DWORD>
This key determines the display configuration – analog-only (VGA), DVO-only, or twin.

SetHead <DWORD>
This key determines the port (analog (VGA) or digital) that controls the timing information. For analog-only (VGA), analog (VGA) should be selected. For DVO-only and twin, digital should be selected.

ConfigId <DWORD>
This key can be used to support multiple configurations and select the configuration. Typically, one configuration is used and this key has a value of 1 to select that configuration.

PowerMgmtShutdownDelay<DWORD>
This key can be used to delay powering down the display when Windows CE is suspended. The value is in milliseconds, and the default value is 0.

AlphaBlendMode<DWORD>

This key is used to enable ARGB32 pixel format on the framebuffer. When in this mode, the alpha channel in the frame buffer is used by the HW internally to blend the graphics plane over the video overlay plane. The value for this registry key can be either 0: disabled, or 1: enabled.

Notes:

- This feature is only supported with the "Depth" registry key set to 0x20 (32). It is ignored otherwise.
- With this mode enabled, any framebuffer pixel with a zero alpha value is transparent – which unfortunately means that the CE desktop and CE utilities will be transparent. Direct Draw must be used to render non-transparent pixels to the framebuffer. If GDI is used, only black will be seen.
- This feature is enabled automatically by the display driver. The driver does this by enabling destination color keying and by configuring the framebuffer to use ARGB32 pixel format. The one side effect this configuration is that any pixel that has an alpha value of 0xFF that also matches the color key selected will be transparent instead of opaque. For this reason, it is recommended to use alpha values from 0x00-0xFE.
- When in this mode, the HW assumes that the framebuffer is pre-multiplied. When using Direct Draw to render to the framebuffer, it is necessary to use the correct pre-multiplied/non-premultiplied flags. If this is done incorrectly, the final rendered image may not look correct. When using the CPU to draw on the framebuffer (ex: SW emulated BLT, memcpy, and so on), you must take into account the fact that the framebuffer is pre-multiplied.

VRWorkaroundMode<DWORD>

This key is used to enable the Video Renderer Workaround that prevents the stock CE 5.0 video renderer from drawing the color key on the frame buffer. This key is ignored if "AlphaBlendMode" is disabled. This key should be enabled whenever "AlphaBlendMode" is enabled and the stock CE 5.0 video renderer is used.

4.1.2 Port Driver (TV Encoder DLL) Options

The port driver (TV encoder DLL) options control the selection of the port driver and the discovery of the TV encoder device.

I2cPin <DWORD>
This setting selects the I^2C* bus. (Value of 1 usually identifies DVO-A, 4 usually identifies DVO-B/C)

I2cDab <DWORD>
This setting selects the Data Address Byte (DAB) for the TV encoder silicon in the system. The DAB is normally specified by the port driver itself, but this setting allows for customization if needed.

PortOrder <STRING>
This value determines the order that the DVO ports and analog (VGA) port will be enumerated. This also controls the order in which the driver will detect TV encoders. This is especially important for DVO-B and DVO-C devices because they share the same I2C bus. The PortOrder string should include the analog (VGA) port number and the single DVO port that is in use. To avoid confusion, do not refer to multiple DVO ports.

The following setting on the 854 specifies to enumerate the DVO-B port (2) and then the analog (VGA) port (5).

"PortOrder"="2500"

The following settings is required for “Shadow” DVO mode on the 854 with the Conexant CX23899 and Silicon Image 9030 HDMI transmitter.

“PortOrder”=”2350”

PortDrivers <STRING>
This value determines the port driver DLLs that are loaded. A port driver is needed for TV encoders and is used when the DigitalPort is set to DVO or Twin. This string should contain a single port driver.

The following setting for the 830 specifies to load the Focus fs454 DLL.

"PortDrivers"="fs454"

The following setting for the 854 specifies to load the Conexant cx873 DLL. Note: The Conexant cx873 DLL supports both the CX25873 and CX25892 TV encoders.

"PortDrivers"="cx873"

To load the Silicon Image 9030 port driver in single DVO mode.
Note: Using the sii9030 port driver in this mode requires a port order of “5300”

“PortDrivers”=”sii9030”

To load the Conexant CX23899 and Silicon Image 9030 HDMI port driver in “Shadow” DVO Mode.

“PortDrivers”=”cx873 sii9030”

DebugPortMessage <DWORD>
This setting is used to turn on or off the debug messages in port driver module.

4.1.3 Memory Management Options

The memory management options control memory management code.

[HKEY_LOCAL_MACHINE\Drivers\Display\Intel]

MaxFbSize <DWORD>
This key controls the size of the frame buffer in bytes.

830, 854: To compute the frame buffer size required for a given resolution, first compute the width in bytes using the width * bytes per pixel. Then, if that isn't a power of two raise it to the next power of two to get the stride. Finally, compute the total size as stride * height. For example:

- 720x480 @32bpp use “1E0000”
- 720x576 @32bpp use “240000”
- 1280x720 @32bpp use “5A0000”
- 720x480 @16bpp use “F0000”
- 720x576 @16bpp use “120000”
- 1920x1080 @ 32bpp use “870000”

MinVidSurfX <DWORD>
This key sets a minimum size in pixels for a video surface in the horizontal direction. The default minimum width is 16. Note: Do not set this lower than 16.

MinVidSurfY <DWORD>
This key sets a minimum size in lines for a video surface in the vertical direction. The default minimum height is 16. Note: Do not set this key to less than 16.

PageReqLimit <DWORD>
This key limits the number of pages (each page holds 4096 bytes) that can be allocated for video memory. Note: This field is typically not modified.

ReservedMemoryBase <DWORD>
This key is only used for static video memory configurations. It determines the location of video memory in the system. This setting is commented out when using dynamic video memory.

ReservedMemorySize <DWORD>
This key is only used for static video memory configurations. It determines the size of video memory, which starts at the location specified in the registry (ReservedMemoryBase). This setting is commented out when using dynamic video memory.

4.1.4 2D Graphics Options

The display driver support options to control 2D rendering. Some of these also affect video rendering.

[HKEY_LOCAL_MACHINE\Drivers\Display\Intel]

SysToVidStretch <DWORD>
This key determines if the system to video stretch BLT functionality is enabled in the driver.

0 = disabled
1 = enabled

BlendFilter <DWORD>
This key determines the blending filter used for 2D stretch BLTs via GDI stretch BLT and DirectDraw AlphaBlt routines. It only affects GDI stretch BLTs when the GDI stretch blit mode is set to BILINEAR through the SetStretchBltMode GDI API. It only affectsAlphaBlt-based stretch blits when the DDABLT_FILTERENABLE flag is used.

1 = Bilinear
2 = Anisotropic

BlitSyncVBI <DWORD>
This key determines if a 2D BLT operation will be blocked until the vertical blanking interval before executing. The BLT occurs during the vertical blanking period, which updates the frame buffer when active data is not being displayed. This can be used to prevent tearing if BLT operations are being requested directly on to the primary frame buffer.

Note: This key enabled for GDI only, not DirectDraw – If DirectDraw is used, double buffering should be used to accomplish this because flipping the back buffer on-screen will sync to the vertical blanking interval automatically. Flipping will utilize less CPU than using BlitSyncVBI because BlitSyncVBI busy waits for the vertical blanking interval within the display driver, whereas flipping is asynchronous.

0 = disabled
1 = enabled

InterlacedToProgressiveAutoBob <DWORD>
This key determines whether hardware bob will be automatically enabled for overlays *if* interleaved (woven) source is provided *and* the GMCH is in a progressive display mode. If this key is disabled, hardware bob will *not* be automatically enabled for overlays.

Note: The driver dynamically checks the state of this registry key value whenever `IDirectDrawSurface5::UpdateOverlay` is called to show the overlay.

Note: This registry key value is only supported on the 830 and 854 GMCH.

0 = disabled
1 = enabled (default)

4.1.5 GDI Options

The following option is a GDI option that indirectly affects the display driver. The display driver does not read this key. Instead, it affects how GDI calls the display driver. For more information, search for "ForceGRAY16" on www.microsoft.com\msdn.

[HKEY_LOCAL_MACHINE\System\GDI]

ForceGRAY16 <DWORD>
This key controls GDI anti-aliasing.

4.1.6 Video Options

The display driver supports a number of options to control video rendering. See the 2D Graphics Options section above for additional options that affect video rendering.

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\DirectX\DirectShow\Video Renderer]

The following settings affect the direct show video renderer.

MaxBackBuffers <DWORD>
This setting determines if single, double, or triple buffering is used. This is the number of back buffers, so please keep in mind that you have a primary buffer as well. Setting this to 0 gives you single buffering, 1 – double buffering, and 2 – triple buffering.

KeyColor <HEX>
This setting configures the color key. This value is in R-G-B format in HEX and corresponds to the color drawn on the frame buffer in areas where the video overlay is to be positioned.

Note: The key can be changed at run-time, but Media Player must be restarted to pick up the change. In cases where this is being used to debug an issue, choose a bright color. The default color tends to be dark grey or black, which can hardly be differentiated from the video image.

4.1.7 Direct3D Mobile Options

Direct 3D Mobile is supported on the 830 and 854.

[HKEY_LOCAL_MACHINE\System\D3DM\Drivers]

RemoteHook <STRING>
This is set to “ddi_iceg” to inform GWES which driver to request Direct3D Mobile support from. Leaving this blank will disable Direct3D Mobile in the operating system.

[HKEY_LOCAL_MACHINE\Drivers\Display\Intel]

Enable3D <DWORD>
This setting enables or disables 3D support in the display driver. 0 disables support, 1 enables support. If it is enabled, a valid Direct3D Mobile needs to be loaded. See 3DModule below.

SharedMemHeapSize <DWORD>

This key is used to set the size (in bytes) of the shared system memory heap to be allocated at driver startup. This setting defaults to 1MB if not specified otherwise.

SharedMemHeapMaxAllocSize <DWORD>
This key is used to set the maximum size (in bytes) to allocate from the shared system memory heap. Any requests larger than this will have a dedicated memory allocation created for them instead of coming from the common heap. This setting defaults to 256KB if not specified otherwise.

[HKEY_LOCAL_MACHINE\Drivers\Display\Intel\3D]

3DModule <STRING>
This setting determines which Direct3D Mobile module gets loaded. Leaving this setting blank will load no Direct3D Mobile module.

4.2 Port Driver (TV Encoder DLL) Attributes

Port driver (TV Encoder DLL) attributes are used if the display type is either DVO-only or Twin. In that case, the display driver reads port driver attribute settings from the registry, tests their values, and if they are valid, passes them to the port driver during initialization. Port driver attributes have Boolean, range, String, or list types. Boolean attributes values must be 0 or 1. Range attributes must match the minimum, maximum, or an internal value computed using the step – similar to a "for" loop in C Language "for (valid = min; valid <= max, valid += step)". String values are typically read-only. List values must match one of the list entries.

If an attribute name does not match a supported attribute or if an out of range value is used, the display driver will fail to load and this will result in a GWES exception.

Port driver attributes are set at the following key:

[HKEY_LOCAL_MACHINE\Drivers\Display\Intel\Param\Attr]

4.2.1 Port Driver Attribute Notes

Note: If an attribute name does not match a supported attribute, or if an out of range value is used, the display driver will fail to load and this will result in a GWES exception.

Note: The port drivers include a TV Format attribute. Please note that changing this attribute at run-time is not supported in the display driver.

4.2.2 Conexant Port Driver Attributes

See the configuration file video.reg for information on the Conexant port driver attributes. Look for details in the section entitled "Conexant CX25873/CX25892 port attributes". Contact Conexant for more information on the Conexant port driver.

4.2.2.1 DAC setting with different TV Output Type on HH2

Currently there is a new video card named “Hominy Hill2” that uses the CX25899 as the TV encoder. There are four analog outputs on that card, which can be configured to different signals in order to adapt to the different TV Output Types. In video.reg, there are 4 parameters for analog output’s setting, “DACA, DACB, DACC, DACD”.

So if the parameter “TV Output” is set to 1 (Composite + S-Video), 2 (Composite) and 3 (S-Video), the DACs need to be set to:

“DACA”=dword:4

“DACB”=dword:3

“DACC”=dword:2

“DACD”=dword:1

Else if the parameter “TV Output” is set to 4 (Component) and 6 (Composite+RGB), the DACs need to be set to:

“DACA”=dword:3

“DACB”=dword:1

“DACC”=dword:2

“DACD”=dword:4

Option 6 of “TV Output” is for a special TV connector named “SCART I” which is very popular in European. This option is only available for the “Hominy Hill2” now since it can output four analog video signals at the same time.

The default DAC setting in video.reg is for “Lost Creek”(CX25892) when you select the "PortDrivers" as "cx873", so please modify this setting if the video card is changed among “Miller Creek”, “Lost Creek” and “Hominy Hill2”.

4.2.3 Focus Port Driver Attributes

See the configuration file video.reg for information on the Focus port driver attributes. Look for details in the section entitled “Focus FS454 port attributes”. Contact Focus for more information on the Focus port driver.

4.2.4 Silicon Image Port Driver Attributes

See the configuration file video.reg for information on the Silicon Image port driver attributes. Look for details in the section entitled “Silicon Image 9030 port attributes”.

5 Configuration

The display driver release includes a video.reg configuration file modified to run with the target platform's default configuration.

5.1 Default Configuration

The configuration file video.reg is designed to work without changes in the following default configurations.

5.1.1 830 Default Configuration

- Platform: Intel® 830M4 Development Platform
- TV Encoder: Spartan-B ADD card with the Focus FS454
- Frame Buffer: 720x480@60Hz and 32 bits per pixel
- TV Format: NTSC
- TV Output Type: Composite and S-Video
- Enabled Ports: DVO (TV) and analog (VGA) with both using the TV encoder timings.
- 3D Support: Disabled

5.1.2 854 Configuration

- Platform: Intel® 854 Development Platform
- TV Encoder: Lost Creek-200 ADD card with the Conexant CX25892 enabled.
- Frame Buffer: 720x480@60Hz and 32 bits per pixel
- TV Format: NTSC
- TV Output Type: Composite and S-Video
- DVI: Enabled
- Enabled Ports: DVO (TV) and analog (VGA) with both using the TV encoder timings.
- 3D Support: Disabled

5.2 Configuration Changes

The following sections describe how to change the video.reg configuration file to accomplish some common configuration changes.

5.2.1 Changing the TV Format to PAL-B

To change the TV format to PAL-B, do the following:

1. First, change the frame buffer to 720x576@50.

- Leave the width at 720
- Change the height setting to 576
  - "Height"=dword:240
- Change the refresh setting to 50
  - "Refresh"=dword:32

2. Next, change the TV format to PAL-B, which is dependent on the TV Encoder.

- On the Focus FS454 use:
  - "TVFormat"=dword:2
- On the Conexant CX25892 use:
  - "TV Format"=dword:4

5.2.2 Changing the TV Output Type to Component

To change the TV output type to Component, do the following:

- On the Focus FS454 use:
  - "TVOutType"=dword:3
- On the Conexant CX25892 use:
  - "TV Output"=dword:4

5.2.3 Enabling 3D Support

To enable 3D support, do the following:

- Enable 3D Support:
  - “Enable 3D”=dword:1
- Load a 3D Module
  - “3DModule”=”gdld3dm”

5.2.4 HDMI Support (Single, “Shadow” DVO Modes)

To enable HDMI support with the Silicon Image 9030 port driver, do the following

Note: “Shadow” DVO works with the following ED/HD resolutions only: 480P, 576P, 720P, and 1080i

- Load the Silicon Image port driver:
  - Single DVO: “PortDrivers”=”sii9030”
  - “Shadow” DVO: “PortDrivers”=”cx873 sii9030”
- Set Port Order:
  - Single DVO “PortOrder”=”5300” ;VGA generates clocks
  - “Shadow” DVO: “PortOrder”=”2350” ;Conexant CX23899 generates clocks
- For both DVO modes Specify Silicon Image i2cpin and i2cdab in DVO-C registry key:
  - “I2cPin”=dword:3
  - “I2cDab”=dword:72
- For “Shadow” DVO mode Conexant I2cPin and I2CDab in DVO-B registry key. Also specify Digital TV output in Conexant Port Attributes
  - “I2cPin”=dword:4
  - “I2cDab”=dword:88
  - “TV Format”=dword:14

6 Hardware Limitations

6.1 830 Chipset Limitations

- YUV 4:2:0 planar format overlay surfaces, that contain interleaved data, are displayed with the temporal chroma artifact, which is commonly referred to as the Chroma Upsampling Error (CUE).
- Both interleaved and progressive data *are* supported by the YUV 4:2:2 packed and xRGB formats.

6.2 854 Chipset Limitations

- YUV 4:2:0 planar format overlay surfaces, that contain interleaved data, are displayed with the temporal chroma artifact, which is commonly referred to as the Chroma Upsampling Error (CUE).
- Both interleaved and progressive data *are* supported by the YUV 4:2:2 packed and xRGB formats.

7 Programming Interface

7.1 WinCE Graphics Driver Escape Interface

The Intel® Digital Set Top Box Graphics Driver for Microsoft Windows CE .NET 4.2/5.0 supports the WinCE system API - ExtEscape(), which provides a dynamic configuration channel to the display driver.

Following shows the ExtEscape API function prototype and the usage module for the user application to communicate with the display driver.

```
ExtEscape(
HDC <Handle to graphic device context>,

int <Escape Code supported by the display driver>
       Note: see below table of the list of the supported escape codes by the display
driver

int <size of the escape code specific input data structure by number of bytes>,

LPCSTR <pointer of input data buffer of BYTE>
   Note: Please note that LPCSTR does not imply the input data buffer is anything
related to "string", which means a long constant pointer of char (BYTE) buffer.  User
need convert the escape code specific input data structure pointer to LPCSTR

int <size of the escape code specific output data structure by number of bytes>,

LPSTR <pointer of input data buffer of BYTE - "Input Data Structure">
       Note: Similar like LPCSTR, LPSTR does not imply the output data buffer is
anything related to "string", which means a long pointer of char (BYTE) buffer. User
need convert the escape code specific output data structure pointer to LPSTR.

);
```

After the Graphics driver is installed, three *.h files (`icegd_public_escape.h`, `igd_pd.h` and `igd_public.h`) can be found under the directory: \WINCE500\IntelCEG\Drivers\Display\I8xx\Extras\Src\Inc. The file video.reg can be found under the directory: \WINCE500\IntelCEG\Drivers\Display\I8xx.

Following is a summary of the escape codes supported by the display driver. Please refer to `icegd_public_escape.h` for more details.

Summary of Supported Graphics Driver Escape Codes

ICEGD_ESCAPE_ENABLE_PORT
ICEGD_ESCAPE_GET_NUM_PD_ATTRIBUTES
ICEGD_ESCAPE_GET_AVAIL_PD_ATTRIBUTES
ICEGD_ESCAPE_SET_PD_ATTRIBUTES
ICEGD_ESCAPE_SET_OVL_COLOR_PARAMS
ICEGD_ESCAPE_SET_FB_GAMMA_RAMP
ICEGD_ESCAPE_GET_MEM_STATS
ICEGD_ESCAPE_GET_MONITOR_DESC
ICEGD_ESCAPE_VBI_ENABLE
ICEGD_ESCAPE_VBI_DISABLE
ICEGD_ESCAPE_VBI_UPDATE
ICEGD_ESCAPE_INTERRUPT_INSTALL
ICEGD_ESCAPE_INTERRUPT_UNINSTALL
ICEGD_ESCAPE_SET_VIDEO_PLANE_ZORDER
ICEGD_ESCAPE_SET_HW_ALPHA_COMPAT_MODE
ICEGD_ESCAPE_DRIVER_VERSION
ICEGD_ESCAPE_I2C_CONFIG
ICEGD_ESCAPE_I2C_ACCESS
ICEGD_ESCAPE_MULT_DVO_SELECT
ICEGD_ESCAPE_QUERY_HDMI_HOTPLUG
ICEGD_ESCAPE_WAIT_HDMI_HOTPLUG
ICEGD_ESCAPE_SET_SCAN_INFO

Note: Please refer to the video.reg (under \WINCE500\IntelCEG\Drivers\Display\I8xx) file that installed with the Intel Display Driver for a list of specific hardware supported. For example, the video.reg file has the data address byte (DAB) required by the I2C bus.

7.1.1.1 ICEGD_ESCAPE_ENABLE_PORT

Input Data Structure	Output Data Structure	Notes
esc_port_ctrl_t	esc_status_t	esc_port_ctrl_t port options: 0 – Primary Display 1 – Secondary Display esc_port_ctrl_t enable options: 0 – disable 1 – enable esc_status_t status options: 0 – ESCAPE_STATUS_NOERROR 1 – ESCAPE_STATUS_ERROR

Description

This escape code is used to enable/disable primary/secondary display port.

Input Data Structure Definition

The data structure **`esc_port_ctrl_t`** should be filled in with port number (0 or 1) and enable (1) /disable (0) value to configure which port (0 or 1) to be enabled or disabled.

Output Data Structure Definition

User can get return success value or error code from **`esc_status_t`**;

Example

```
esc_port_ctrl_t port;
memset(&port, 0, sizeof(esc_port_ctrl_t));
//choose primary display
port.port = 0;
//enable display port 0
port.enable = 1;

esc_status_t status;
memset(&status, 0, sizeof(esc_status_t));

ExtEscape(
    Hdc,
    ICEGD_ESCAPE_ENABLE_PORT,
    sizeof(esc_port_ctrl_t),
    (LPCSTR)&port,
    sizeof(esc_status_t),
    (LPSTR)&status);
// check the status here for error handling
// …
```

7.1.1.2 ICEGD_ESCAPE_GET_NUM_PD_ATTRIBUTES

Input Data Structure	Output Data Structure	Notes
N/A	int	Must be used before a call to ICEGD_ESCAPE_GET_AVAIL_PD_ATTRIBUTES to size the required igd_attr_t data structure array.

Description

This escape code is used to request the total number of attributes supported by the Port Driver, which will be used by ICEGD_ESCAPE_GET_AVAIL_PD_ATTRIBUTES to calculate the size of the buffer for the attribute list.

Example

See ICEGD_ESCAPE_SET_PD_ATTRIBUTES for a complete example.

7.1.1.3 ICEGD_ESCAPE_GET_AVAIL_PD_ATTRIBUTES

Input Data Structure	Output Data Structure	Notes
N/A	A buffer pointed by igd_attr_t*	All structures below should be with same size and should be converted to others according to the attribute type. igd_attr_t: General attribute structure igd_range_attr_t: Range type attribute structure igd_list_attr_t: List type attribute igd_list_entry_attr_t : Entry for a list igd_bool_attr_t: Boolean type attribute igd_string_attr_t: String type attribute Dependent on previous call to ICEGD_ESCAPE_GET_NUM_PD_ATTRIBUTES to determine size of output buffer required

Description

This escape code is used to get available attributes exported by the Port Driver. ICEGD_ESCAPE_GET_NUM_PD_ATTRIBUTES must be called before it can get the number of attributes. The output is a pointer to a list that can hold all the attributes supported by the Port Driver.

It is always called to get the attribute list of the old configuration for the later use by ICEGD_ESCAPE_SET_PD_ATTRIBUTES.

Output Data Structure Definition

`igd_attr_t` is general envelop data structure for all type of attributes. It supports the following five different types:

- range
- list
- list entry
- bool
- string

There are five **`igd_<type>_attr_t`** data structures with the same size of structure **`igd_attr_t`** (where <type> is one of the above types). The user can generally use the **igd_attr_t** pointer to any attribute data buffer and access the interested member data by the specific **`igd_<type>_attr_t`** data structure, as listed below.

- igd_*range*_attr_t
- igd_*list*_attr_t
- gd_*list*_entry_attr_t
- igd_*bool*_attr_t
- igd_*string*_attr_t;

Please refer to `igd_pd.h` for attribute data structure details.

Example

See ICEGD_ESCAPE_SET_PD_ATTRIBUTES for a complete example.

7.1.1.4 ICEGD_ESCAPE_SET_PD_ATTRIBUTES

Input Data Structure	Output Data Structure	Notes
igd_attr_t	N/A	Only a single attribute can be set per call.

Description

This escape code is used to set Port Driver attributes. It is always called after the attribute list filled by ICEGD_ESCAPE_GET_AVAIL_PD_ATTRIBUTES.

The attribute list is different between different cards. If you want to make sure whether the attribute could be supported by the card, you can use ICEGD_ESCAPE_GET_AVAIL_PD_ATTRIBUTES to get the supported attribute list and check with it.

String attributes are read only.

Example

```
Please refer to the following instructions for modify example code below for other PD
attribute types at the place holder with the marks about // (1) (2) (3)

/* if the type is PD_ATTR_TYPE_LIST, the lines with (//(1), (2) and (3)) should be
replaced as below:
         igd_list_attr_t *list;                                              (1)
         list = (igd_list_attr_t *) current;                        (2)
         list->current_index = 2;                                    (3)

   if the type is PD ATTR TYPE BOOL, the lines with (//(1), (2) and (3)) should be
replaced as below:
         igd_bool_attr_t *bool;                                          (1)
     bool = (igd bool attr t *) current;                    (2)
     bool->current_value = arg_attr_value;              (3)
*/
```

```
int number;//available attributes number
igd attr t *attr list, *current;
/*other types such as igd list attr t, igd bool attr t, igd string attr t
can refer to the implement of igd range attr t as below*/
igd range attr t *range;                                    //(1)


esc status t status;

ExtEscape(
   Hdc,
       ICEGD_ESCAPE_GET_NUM_PD_ATTRIBUTES,
      0,
      NULL,
      sizeof(unsigned int),
  (LPSTR)&number);

//the attribute list should contain number entries of igd attr t.
attr list = (igd attr t *) calloc(number, sizeof(igd attr t));

//now the attr list is the old configure of port driver
ExtEscape(
   Hdc,
   ICEGD_ESCAPE_GET_AVAIL_PD_ATTRIBUTES,
   0,
   NULL,
   number * sizeof(igd_attr_t),
   (LPSTR) attr_list);

current = attr_list;
for (int i=0; i<number; i++, current++)
{
   //choose attribute DACA, and the type of it must be range
   if (current->id == 26 && current->type == PD_ATTR_TYPE_RANGE)
   {
       /*to set the value must force the structure convert from igd_attr_t
         to igd_range_attr_t */
       range = (igd_range_attr_t *) current;
//(2)
          range->current_value = 2;
//(3)

       memset(&status, 0, sizeof(esc status t));

       ExtEscape(
          Hdc,
          ICEGD ESCAPE SET PD ATTRIBUTES,
          sizeof(igd attr t),
          (LPCSTR) current,
          sizeof(esc status t),
          (LPSTR) &status);
       break;
   }
       /*if type is list, there will follow some entries, the number of which
         is stored in the num entries of the structure. So the pointer to the
         list should jump num entries to find next attribute*/
   if (current->type == PD ATTR TYPE LIST) {
       i+= ((igd list attr t *)current)->num entries;
       current += ((igd list attr t *)current)->num entries;
   }
}
free(attr list);
```

7.1.1.5 ICEGD_ESCAPE_SET_OVL_COLOR_PARAMS

Input Data Structure	Output Data Structure	Notes
esc_color_params_t	N/A	esc_color_params flag options: Optional: GAMMA_FLAG (0x1) BRIGHTNESS_FLAG (0x2) CONTRAST_FLAG (0x4) SATURATION_FLAG (0x8) Mandatory: OVL_COLOR_FLAG (0x10)

Description

This escape code is used to set overlay attributes (brightness, contrast, saturation, and gamma). Please make sure that OVL_COLOR_FLAG should always be set. The sample affect only when using overlay.

Input Data Structure Definition

The data structure **`esc_color_params_t`** should be filled in with parameters of gamma, brightness, contrast, saturation and so on. Please refer to `icegd_public_escape.h` for details.

Example

```
esc color params t color;

memset(&color, 0, sizeof(esc color params t));
//right now we change only brightness and contrast
color.flag = OVL COLOR FLAG | BRIGHTNESS FLAG | CONTRAST FLAG;
color.brightness = 0x0;
color.contrast = 0x0;

for(int i=0;i<20;i++){
//change the brightness and contrast gradually
    ExtEscape(
       Hdc,
       ICEGD ESCAPE SET OVL COLOR PARAMS,
       sizeof(esc color params t),
       (LPCSTR)&color,
       0,
               NULL);
   //Check error status returned by ExtEscape
   // …
   // Wait a while for HW to complete the operation of changing the overlay color
parameters
       Sleep(500);
       color.brightness+=3000;
       color.contrast+=3000;
}
```

7.1.1.6 ICEGD_ESCAPE_SET_FB_GAMMA_RAMP

Input Data Structure	Output Data Structure	Notes
esc_color_params_t	N/A	esc_color_params flag options: Mandatory: GAMMA_FLAG (0x1) Not Applicable: BRIGHTNESS_FLAG (0x2) CONTRAST_FLAG (0x4) SATURATION_FLAG (0x8) OVL_COLOR_FLAG (0x10)

Description

This escape code is used to set frame buffer gamma correction ramp. Legal values are from 1-500, 1 - default, 500 – brightest.

Input Data Structure Definition

Same data structure as ICEGD_ESCAPE_SET_OVL_COLOR_PARAMS, but for this escape code, you need only set the GAMMA_FLAG.

Example

```
esc_color_params_t color;
memset(&color, 0, sizeof(esc_color_params_t));

//set flag to gamma
color.flag = GAMMA_FLAG;
color.gamma = 0;

for(int i=1;i<10;i++){
    //change gamma value gradually, and the legal value is between 1 and 500
    color.gamma=i*50;
    ExtEscape(
       Hdc,
       ICEGD_ESCAPE_SET_FB_GAMMA_RAMP,
       sizeof(esc_color_params_t),
       (LPCSTR)&color,
       0,
   NULL);
//Check error status returned by ExtEscape
// …
// Wait a while for HW to complete the operation of changing the gamma value
    Sleep(500);
}
```

7.1.1.7 ICEGD_ESCAPE_GET_MEM_STATS

Input Data Structure	Output Data Structure	Notes
N/A	esc_mem_stats_t	

Description

This escape code is used to get memory statistics.

Output Data Structure Definition

Please refer to igd_public.h for details about data structure **`esc_mem_stats_t`**.

Example

```
esc_mem_stats_t stats;
memset(&stats, 0, sizeof(esc_mem_stats_t));

ExtEscape(
    Hdc,
    ICEGD_ESCAPE_GET_MEM_STATS,
    0,
    NULL,
    sizeof(esc_mem_stats_t),
    (LPSTR) & stats);
```

7.1.1.8 ICEGD_ESCAPE_GET_MONITOR_DESC

Input Data Structure	Output Data Structure	Notes
Int	esc_monitor_desc_t	The input is port number, which can be only 0 or 1.

Description

This escape code is used to get the monitor description (EDID).

Output Data Structure Definition

Please refer to `icegd_public_escape.h` for details about data structure **`esc_monitor_desc_t`**.

Example

```
int port = 0;
esc_monitor_desc_t esc_monitor_desc;
memset(&esc_monitor_desc, 0, sizeof(esc_monitor_desc_t));

ExtEscape(
    Hdc,
    ICEGD_ESCAPE_GET_MONITOR_DESC,
    sizeof(port),
    (LPCSTR) &port,
    sizeof(esc_monitor_desc_t),
    (LPSTR) &esc_monitor_desc);
```

7.1.1.9 ICEGD_ESCAPE_VBI_ENABLE

Input Data Structure	Output Data Structure	Notes
N/A	N/A	

Description

This escape code is used to call port driver to enable Line 21 VBI (Vertical Blank Interrupt) data. It is always used before ICEGD_ESCAPE_VBI_UPDATE with ICEGD_ESCAPE_VBI_DISABLE following.

Example

```
ExtEscape(
    Hdc,
    ICEGD_ESCAPE_VBI_ENABLE,
    0,
    NULL,
    0,
    NULL
    );
```

7.1.1.10 ICEGD_ESCAPE_VBI_DISABLE

Input Data Structure	Output Data Structure	Notes
N/A	N/A	

Description

This escape code is used to call port driver to disable Line 21 VBI (Vertical Blank Interrupt) data. It is always used after ICEGD_ESCAPE_VBI_UPDATE finished.

Example

```
ExtEscape(
    Hdc,
    ICEGD_ESCAPE_VBI_DISABLE,
    0,
    NULL,
    0,
    NULL
);
```

7.1.1.11 ICEGD_ESCAPE_VBI_UPDATE

Input Data Structure	Output Data Structure	Notes
vbi_data	N/A	

Description

This escape code is used to call port driver to update VBI. It passes VBI line information to the TV encoder to insert into the analog TV signal. Before you update VBI, you should enable VBI, and disable VBI after the update has completed. There should be a short wait between every update operation.

Input Data Structure Definition

Please refer to igd_public.h for **vbi data** structure details.

Example

```
vbi_data*   pvbi = 0;
pvbi = (vbi_data*)malloc( sizeof(vbi_data) + 2 );
pvbi->unFlags = VERSION_1;
pvbi->unDataLength = 2;
/* CC, CGMS  (line 21)  */
pvbi->unDataType = LINE_21_FIELD_1;

//enable vbi
ExtEscape(
    Hdc,
    ICEGD_ESCAPE_VBI_ENABLE,
    0,
    NULL,
    0,
    NULL
    );

//selects paint-on style
pvbi->pucData[ 0 ] = 0x14;
pvbi->pucData[ 1 ] = 0x29;
ExtEscape(
    Hdc,
    ICEGD_ESCAPE_VBI_UPDATE,
    sizeof(vbi_data) + 2,
    (const char*)pvbi,
    0,
    NULL);
//33 is the time between byte updates. TV seems unable to handle it too fast.
Sleep(33);

//select position and color(green)
pvbi->pucData[ 0 ] = 0x11;
pvbi->pucData[ 1 ] = 0x62;
ExtEscape(
    Hdc,
    ICEGD_ESCAPE_VBI_UPDATE,
    sizeof(vbi_data) + 2,
    (const char*)pvbi,
    0,
    NULL);
Sleep(33);

//display a string "abababababab"
for(int i=0;i<5;i++){
    pvbi->pucData[ 0 ] = 'a';
    pvbi->pucData[ 1 ] = 'b';
```

```
    ExtEscape(
        Hdc,
        ICEGD_ESCAPE_VBI_UPDATE,
        sizeof(vbi_data) + 2,
        (const char*)pvbi,
        0,
        NULL);
    Sleep(33);
}
Sleep(1000);

//erase cc displayed memory
pvbi->pucData[ 0 ] = 0x14;
pvbi->pucData[ 1 ] = 0x2c;
ExtEscape(
    Hdc,
    ICEGD_ESCAPE_VBI_UPDATE,
    sizeof(vbi_data) + 2,
    (const char*)pvbi,
    0,
    NULL);
Sleep(33);

//disable vbi
ExtEscape(
    Hdc,
    ICEGD_ESCAPE_VBI_DISABLE,
    0,
    NULL,
    0,
    NULL
    );

free( pvbi );
```

7.1.1.12 ICEGD_ESCAPE_INTERRUPT_INSTALL

Input Data Structure	Output Data Structure	Notes
escape_interrupt_data	N/A	ESCAPE_INTERRUPT_VBLANK = 0x1000

Description

This escape code is used to install an interrupt. It is always followed by ICEGD_ESCAPE_INTERRUPT_UNINSTALL. unInterrupt of the input structure could only be ESCAPE_INTERRUPT_VBLANK. After the operation finishes, hWaitObject will return a handle to the OS dependent wait object.

Input Data Structure Definition

Please refer icegd_public_escape.h for details about **`escape_interrupt_data`**;

Example

See ICEGD_ESCAPE_ INTERRUPT_UNINSTALL for a complete interrupt example.

7.1.1.13 ICEGD_ESCAPE_INTERRUPT_UNINSTALL

Input Data Structure	Output Data Structure	Notes
Escape_interrupt_data	N/A	

Description

This escape code is used to un-install an interrupt.

Example

```
escape_interrupt_data interrupt;
memset(&interrupt, 0, sizeof(escape_interrupt_data));
interrupt.unInterrupt = ESCAPE_INTERRUPT_VBLANK;
 //install an interrupt
ExtEscape(
    Hdc,
    ICEGD_ESCAPE_INTERRUPT_INSTALL,
    sizeof(escape_interrupt_data),
    (LPCSTR)&interrupt,
    0,
    NULL
    );

 //if an interrupt occurs, catch it and inform user
if(WaitForSingleObject(interrupt.hWaitObject, 2000)==WAIT_OBJECT_0){
    printf("Get a interrupt!\n");
}

 //unstall the interrupt
ExtEscape(
    Hdc,
    ICEGD_ESCAPE_INTERRUPT_UNINSTALL,
    sizeof(escape_interrupt_data),
    (LPCSTR)&interrupt,
    0,
    NULL
    );
```

7.1.1.14 ICEGD_ESCAPE_SET_VIDEO_PLANE_ZORDER

Input Data Structure	Output Data Structure	Notes
esc_vid_plane_zorder_t	N/A	IGD_ZORDER_OVERLAY_ABOVE_PLANE 0x0 IGD_ZORDER_OVERLAY_BELOW_PLANE 0x1

Description

This escape code is used to set Z-order and key color for plane based alpha support.

Input Data Structure Definition

Please refer to icegd_public_escape.h for details about **esc_vid_plane_zorder_t**;

Example

```
static unsigned long   lcolorkey = 0x090900;
static unsigned long   lvideo plane position = IGD ZORDER OVERLAY BELOW PLANE;
esc vid plane zorder t     vid plane zorder;
memset(&vid plane zorder, 0, sizeof(esc vid plane zorder t));

vid plane zorder.ulKeyColor = lcolorkey;
vid plane zorder.ulZOrder = lvideo plane position;

ExtEscape(
   Hdc,
   ICEGD ESCAPE SET VIDEO PLANE ZORDER,
   sizeof(esc vid plane zorder t),
   (LPCSTR)&vid plane zorder,
   0,
   NULL);
```

7.1.1.15 ICEGD_ESCAPE_SET_HW_ALPHA_COMPAT_MODE

Input Data Structure	Output Data Structure	Notes
Int	N/A	Input can be only : HW_ALPHA_COMPAT_VIDEO_RENDERER_MODE 0x00000001

Description

This escape code is used to enable the HW plane alpha mode.

Example

```
int mode;
mode = HW ALPHA COMPAT VIDEO RENDERER MODE;
ExtEscape(
    Hdc,
    ICEGD ESCAPE SET HW ALPHA COMPAT MODE,
    sizeof(char),
    (LPCSTR)&mode,
    0,
    NULL);
```

7.1.1.16 ICEGD_ESCAPE_DRIVER_VERSION

Input Data Structure	Output Data Structure	Notes
N/A	esc_driver_version_t	

Description

This escape code is used to get the driver version.

Input Data Structure Definition

Please refer to icegd_public_escape.h for details about **esc_driver_version_t**;

Example

```
esc_driver_version_t driver_version;
memset(&driver_version, 0, sizeof(esc_driver_version_t));

ExtEscape(
    Hdc,
    ICEGD_ESCAPE_DRIVER_VERSION,
    0,
    NULL,
    sizeof(esc_driver_version_t),
    (LPSTR) &driver_version);
```

7.1.1.17 ICEGD_ESCAPE_I2C_CONFIG

Input Data Structure	Output Data Structure	Notes
esc_i2c_config_t	N/A	

Description

This escape code is used before ICEGD_ESCAPE_I2C_ACCESS to configure Pin Pair, DAB and I2C speed.

Example

See ICEGD_ESCAPE_I2C_ACCESS for a complete I2C example.

7.1.1.18 ICEGD_ESCAPE_I2C_ACCESS

Input Data Structure	Output Data Structure	Notes
esc_i2c_access_t	unsigned char	In case of read operation, data is returned in output buffer. Write operation data is retrieved from the esc_i2c_access_t input buffer.

Description

This escape code is used to access I2C. It can implement a read or write operation by setting the mode.

Input Data Structure Definition

Please refer icegd_public_escape.h for details about **esc_i2c_access_t**;

Example

```
esc_i2c_config_t i2c_config;
esc_i2c_access_t i2c_access;
unsigned char data;
unsigned char data2;

memset(&i2c_config, 0, sizeof(esc_i2c_config_t));
memset(&i2c_access, 0, sizeof(esc_i2c_access_t));

//the data is only for cx892.
// please refer to special documents for different cards
i2c_config.bus = 4;
i2c_config.dab = 0x88;
i2c_config.speed = 200;

/* Configure I2C Bus access */
ExtEscape(
    Hdc,
    ICEGD_ESCAPE_I2C_CONFIG,
    sizeof(esc_i2c_config_t),
    (LPCSTR)&i2c_config,
    0,
    NULL);

i2c_access.addr = 0x54;
i2c_access.mode = INTEL_I2C_MODE_READ;

/* Read a character from address 0x54 */
ExtEscape(
    Hdc,
    ICEGD_ESCAPE_I2C_ACCESS,
    sizeof(esc_i2c_access_t),
    (LPCSTR)&i2c_access,
    sizeof(unsigned char),
    (LPSTR)&data);

/* Write data 140 back */
i2c_access.mode = INTEL_I2C_MODE_WRITE;
i2c_access.data = 140;
ExtEscape(
    Hdc,
    ICEGD_ESCAPE_I2C_ACCESS,
    sizeof(esc_i2c_access_t),
    (LPCSTR)&i2c_access,
    sizeof(unsigned char),
    (LPSTR)&data);
printf("%d\n",data);
```

```
/* Read the data to make sure the last step works ok */
i2c_access.mode = INTEL_I2C_MODE_READ;
ExtEscape(
    Hdc,
    ICEGD_ESCAPE_I2C_ACCESS,
    sizeof(esc_i2c_access_t),
    (LPCSTR)&i2c_access,
    sizeof(unsigned char),
    (LPSTR)&data2);
printf("%d\n", data2);
```

7.1.1.19 ICEGD_ESCAPE_MULT_DVO_SELECT

Input Data Structure	Output Data Structure	Notes
Esc_mult_dvo_select_t	N/A	@select: ESC_MULT_DVO_MASTER : multiplex DVO master port (DVOB) ESC_MULT_DVO_SLAVE : slave port (DVOC) @flag: User need to set additional flag for the HDMI output on SD ESC_DISPLAY_PIXEL_DOUBLE ESC_DISPLAY_INTERLACED

Description

This escape code is used to select a DVO port (master/slave).

Input Data Structure Definition

Set *select* to select a DVO port. Set *flag* to set double pixel and interlace when HDMI (slave DVO port) is selected under SD mode.

Example

```
esc_mult_dvo_select_t dvoSelect;
// Select slave DVO port (HDMI)
 dvoSelect.select = ESC_MULT_DVO_SLAVE;
dvoSelect.flag   = ESC_DISPLAY_PIXEL_DOUBLE | ESC_DISPLAY_INTERLACED;
 ExtEscape(
     hDC,
     ICEGD_ESCAPE_MULT_DVO_SELECT,
     sizeof(esc_mult_dvo_select_t),
     (LPCSTR)&dvoSelect,
     0,
     NULL
     );
```

7.1.1.20 ICEGD_ESCAPE_QUERY_HDMI_HOTPLUG

Input Data Structure	Output Data Structure	Notes
N/A	esc_enum_hotplug_status	ESC_HDMI_PLUG_IN : ESC_HDMI_PLUG_OUT

Description

This escape code is used to query the HDMI hotplug status.

Output Data Structure Definition

If the HDMI is plugged in, return ESC_HDMI_PLUG_IN, else, return ESC_HDMI_PLUG_OUT.

Example

```
esc enum hotplug status        status ;
 ExtEscape(
     hDC,
     ICEGD ESCAPE QUERY HDMI HOTPLUG,
     0,
     NULL,
     sizeof(esc enum hotplug status) ,
     (LPSTR)&status
     );
 if(status == ESC HDMI PLUG IN)
      tprintf( T(" Status: IN\n"));
 else if(status == ESC HDMI PLUG OUT)
      tprintf( T(" Status: OUT\n"));
 else
     _tprintf(_T(" Invalid Hotplug Status.\n"));
```

7.1.1.21 ICEGD_ESCAPE_WAIT_HDMI_HOTPLUG

Input Data Structure	Output Data Structure	Notes
N/A	esc_enum_hotplug_status	ESC_HDMI_PLUG_IN : ESC_HDMI_PLUG_OUT

Description

This escape code is used to wait for an HDMI hotplug event.

Output Data Structure Definition

Block the user thread until the HDMI is plugged in, then return ESC_HDMI_PLUG_IN, or plugged out, then return ESC_HDMI_PLUG_OUT.

Example

```
  esc_enum_hotplug_status status ;
    esc_mult_dvo_select_t   dvoSelect;

    // Wait for HDMI hotplug event
    ExtEscape(
        hDC,
        ICEGD_ESCAPE_WAIT_HDMI_HOTPLUG,
        0,
        NULL,
        sizeof(esc_enum_hotplug_status) ,
        (LPSTR)&status
        );
    if(status == ESC_HDMI_PLUG_IN){
        dvoSelect.select = ESC_MULT_DVO_SLAVE;
        // Double pixel and interlaced flag only be needed for select HDMI (slave)
under SD mode.
    dvoSelect.flag   = ESC_DISPLAY_PIXEL_DOUBLE|ESC_DISPLAY_INTERLACED;
        }
   else if(status == ESC_HDMI_PLUG_OUT) {
        dvoSelect.select = ESC_MULT_DVO_MASTER;
        }
    else {
        _tprintf(_T(" Invalid Hotplug Status.\n"));
        exit(0);
        }
    // Select the right DVO port
    ExtEscape(
        hDC,
        ICEGD_ESCAPE_MULT_DVO_SELECT,
        sizeof(esc_mult_dvo_select_t),
        (LPCSTR)&dvoSelect,
        0,
        NULL
        );
```

7.1.1.22 ICEGD_ESCAPE_SET_SCAN_INFO

Input Data Structure	Output Data Structure	Notes
Int	N/A	N/A

Description

This escape code is used to set the HDMI scan info.

Input Data Structure Definition

The input data can be set between 0-3. But only 1 and 2 make sense now. 1—overscanned, 2—underscanned.

Example

```
unsigned int num;
num = 1;

ExtEscape(
     hDC,
     ICEGD_ESCAPE_SET_SCAN_INFO,
     sizeof(unsigned long),
     (LPCSTR)&num,
     NULL,
     NULL
);
```