Intel Video Driver Solution for XP Embedded \ WES2009 Running on Intel® Atom™ Platforms

By Sean D. Liming and John R. Malin
Annabooks

June 2011

Intel going Embedded

The Intel® Atom™ processor has become a dominate processor / platform solution in the embedded PC market. The low power, great performance options, and low cost allow companies to use high end operating systems like Windows® XP and Windows® 7 in their designs, where .NET applications and graphical front ends provide a richer user experience. To help developers get the most out of the Atom processors, Intel has developed compilers, diagnostic tools, and optimized libraries.

XP Embedded \ WES2009 is a componentized version of Windows XP Pro for embedded systems. Developers can create components for drivers and applications, and add them to a database of 15,000 components to build custom images. Creating these components can be a challenge. In some cases, you can create the component correctly and the driver doesn't work once the operating systems has completed first boot agent (FBA). In this article, we will look at the issues and solution for creating a component for the Intel chipset video drivers.

A Few Video Problems and the Solution

Over the past few years, we have seen different WES 2009 and WES 7 projects that use the Atom processor such as thin clients, medical test equipment, tablet PCs, gaming systems, point of sale systems, home entertainment systems, and industrial controls. In the course of these projects, we ran into some unexpected video issues. The normal WES 2009 development process includes creating a component based on the generic video driver provided by the manufacturer. This has become a challenge over the years as some manufactures build in limitations that force developers to install their drivers rather than componentize them. Either direction will work; but, as we will see, creating a component leads to a more automated build procedure. The three common video drivers on the market are from NVIDIA, ATI, and Intel. Each presents some interest quirks. Because the Intel Atom chipsets include a video graphics engine, we have implemented driver support for these chipsets quite often, and have run into some interesting problems:

- One of two Video drivers not loading – The Intel chipsets support two video outputs. One for flat panels and the other for VGA or DVO. The generic driver almost loads correctly, but the second video port doesn't. The quick solution is to uninstall the failed driver and then do a scan so Windows can re-load / reinstall the missing driver. This adds an undesirable manual step to a build process that one would rather have automated.

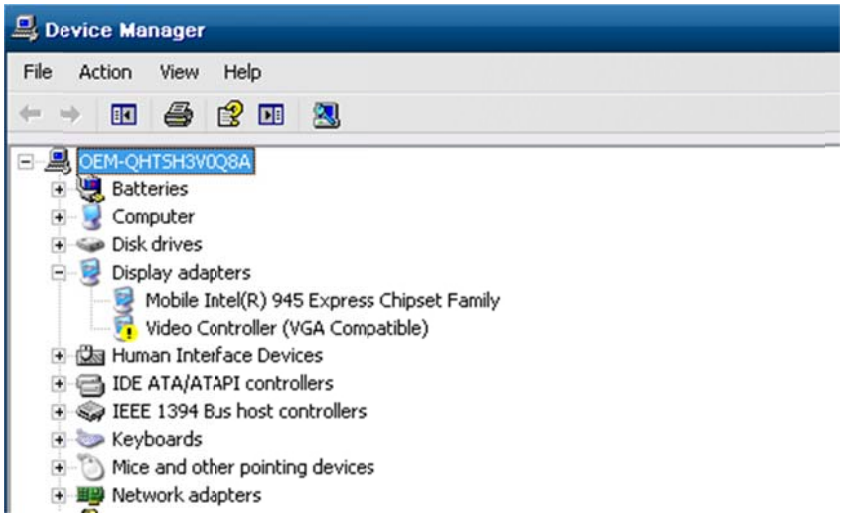

Figure 1 – Generic Video Driver Not Loading During FBA

- LVDS set as primary when only VGA output is available – The build process completes and when the system reboots the VGA screen is blank. Windows is operating normally, but the generic video driver thinks the primary display is the flat panel attached to the LVDS.

- LVDS is primary. VGA is a clone, but not working properly - This last scenario is when the primary display is the flat panel and the device will occasionally be connected to a VGA display. The generic driver doesn't sync to the external display.

One could bounce off the walls trying to figure out why these problems are occurring. You realize the generic driver doesn't provide everything needed. Anticipating future mobile solutions with multiple video outputs and different LCD panels, Intel developed a nice solution to these problems in the form of the Intel® Embedded Graphics Driver (IEGD) and the Intel® Embedded Media and Graphics Driver (EMGD). These two solutions come with a configuration editor that allows you to preset how the driver behaves in the system and lets you set up LCD panel timing for LCD panels that don't support the Extended Display Identification Data standard.

These driver tools can be downloaded from the Intel Embedded Design Center (EDC) – http://edc.intel.com. Each driver supports different chips sets, so you should download the appropriate driver and version for your system. The following table lists the embedded driver solutions and the video chipsets they support. The table is a sample so please visit the EDC website for the official and updated list of supported chipsets and operating systems.

IEGD	EMGD
Intel® Atom™ processor D400/500, N400/500 series	Intel® System Controller Hub US15W, US15WP and US15WPT
Intel® Atom™ processor Z500 series with Intel® System Controller Hub US15W/US15WP/US15WPT	Intel® Atom™ Processor E6xx Series
Intel® Atom™ Processor N270 with Mobile Intel® 945GSE Express Chipset	
Mobile Intel® 945GME Express Chipset	
Intel® 945G Express Chipset	
Intel® Q45, G41 and G45 Express chipsets	
Mobile Intel® GM45, GL40 and GS45 Express chipsets	
Intel® Q35 Express Chipset	
Intel® Q965 Express Chipset	
Mobile Intel® 910GMLE and 915GME Express chipsets	
Intel® 915GV Express Chipset	
Mobile Intel® GME965 and GLE960 Express chipsets	

These drivers solve the problems of the generic driver:

- Drivers load correctly in XP Embedded / WES2009 – A component can be created from the driver's INF file. The driver loads both ports correctly.

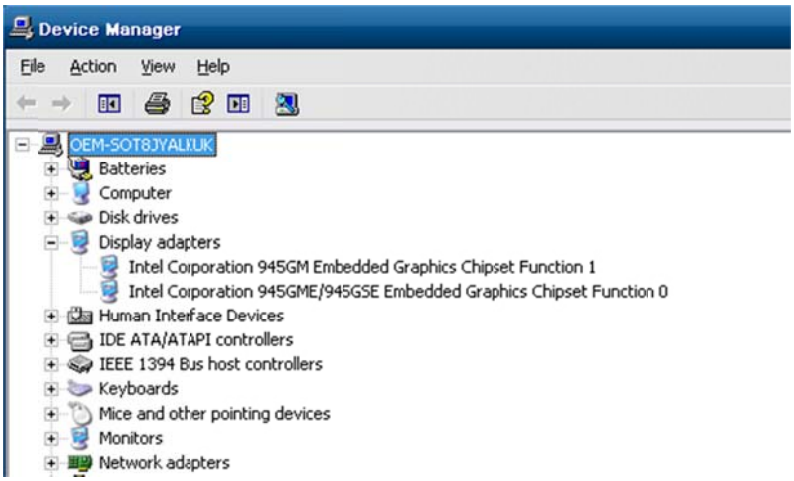

Figure 2 - IEGD Loads Correctly During FBA

- Define the default output – The configuration editor can be used to define what output ports are available and list them in order.

- Define cloning, single, extended support using the configuration editor – The embedded drivers are optimized and have well defined support for the different output options available.

Using IEGD to Create Custom Video Driver Support

The configuration editor supplied with both driver solutions provides the means to preconfigure the driver for LCD panels and port order. There are many options available; but for most developers, the process is very simple. As an example, let's use the editor to create a custom driver for a 945GM chipset that has an LVDS (LCD) as the primary monitor and VGA as a secondary clone output.

The EDC will walk your through the process to download the correct package for your chipset. After you have downloaded and install the IEGD, a shortcut to the configuration editor will be placed on the desktop.

1. Open the IEGD Configuration Editor. The editor includes a quick start guide to help you through the process. The steps in this article are for a specific solution. We recommend that you review the user guide and quick start guide for more details on the different options.

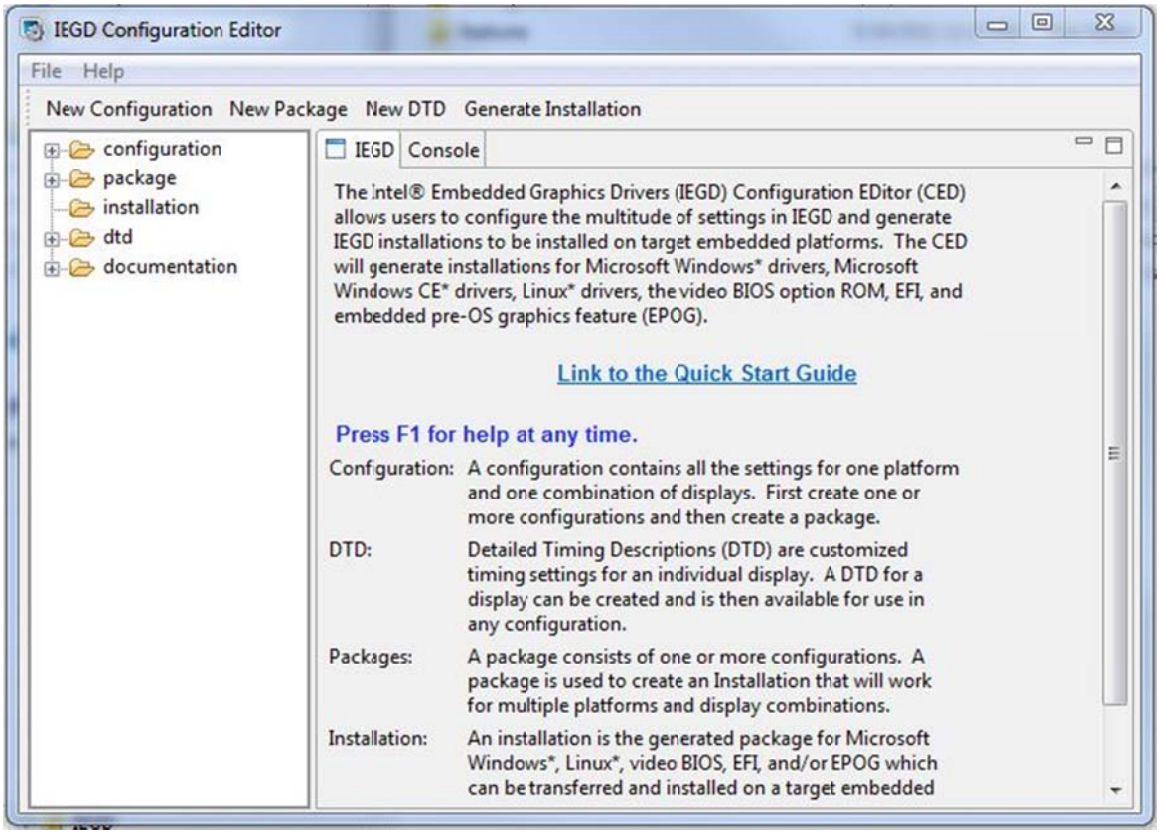

Figure 3 - IEGD Configuration Editor

We will assume that the panel has a standard timing configuration so we can skip the steps to setup LCD panel timing.

2. Click on **New Configuration** from the menu. The Chipset configuration page appears. This page lets you select the chipset, preset the display mode, and set the port order.

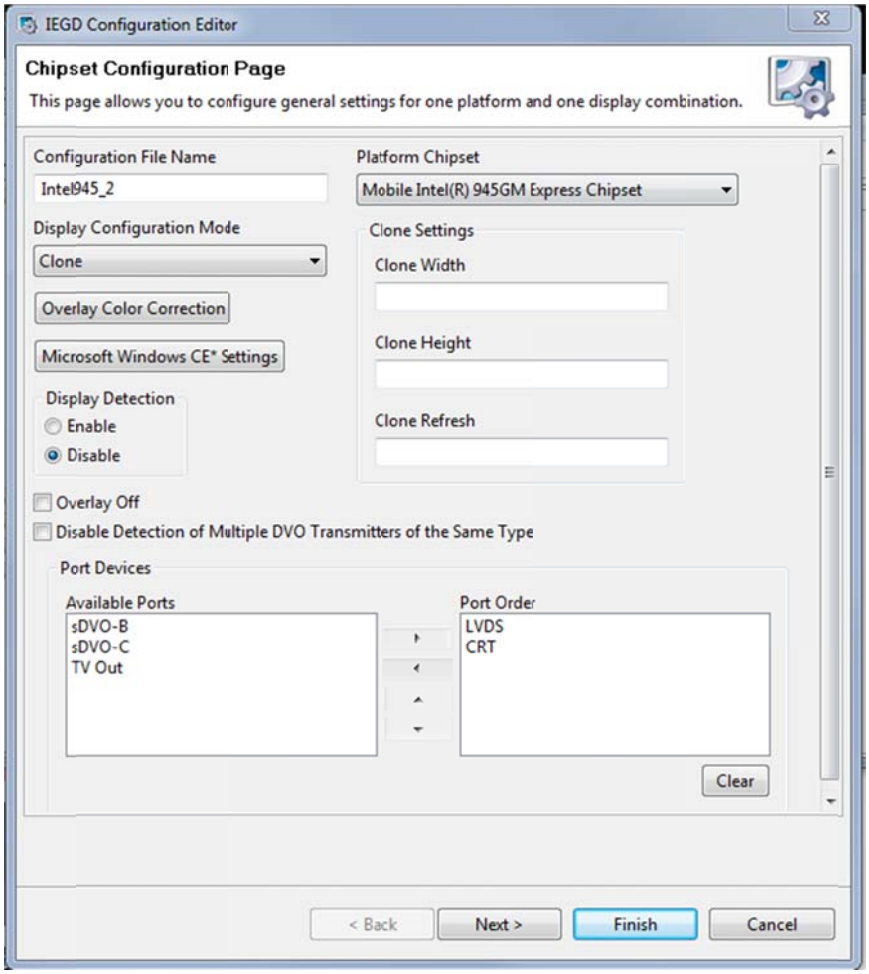

Figure 4 - Configuration Page

3. For this example the following options are set:

   - Configuration Name: **Intel945_2**
   - Platform Chipset: **Mobile Intel® 945GM Express Chipset**
   - Display Configuration Mode: **Clone**
   - Port Devices: select **LVDS** first and then **VGA**. This will make LVDS the primary display and the VGA will be the clone. If you want the VGA to be the primary you can change the order of the ports.

4. Click **Next**. Each port will need to be configured. The first is the LVDS.

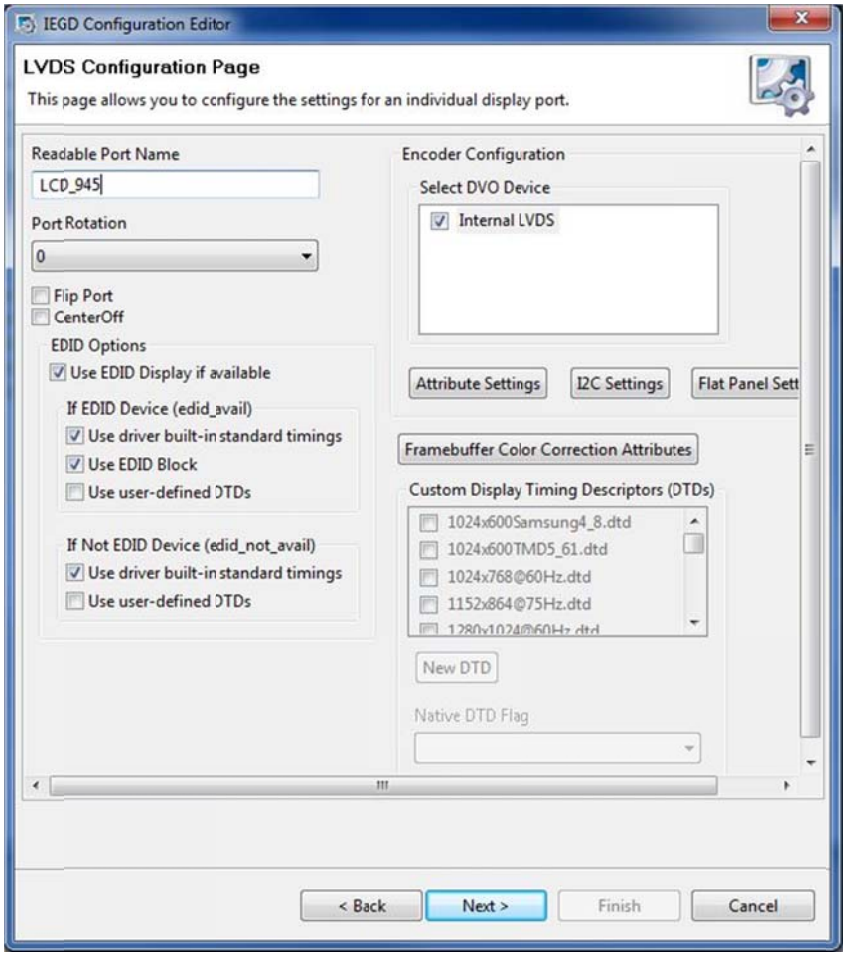

Figure 5 - Settings for the LVDS Port

5. Enter the following for LVDS:

   - Readable Port Name: **LCD_945**
   - Select DVO Device: **Internal LVDS**

6. Click **Next**. The VGA port configuration page appears.

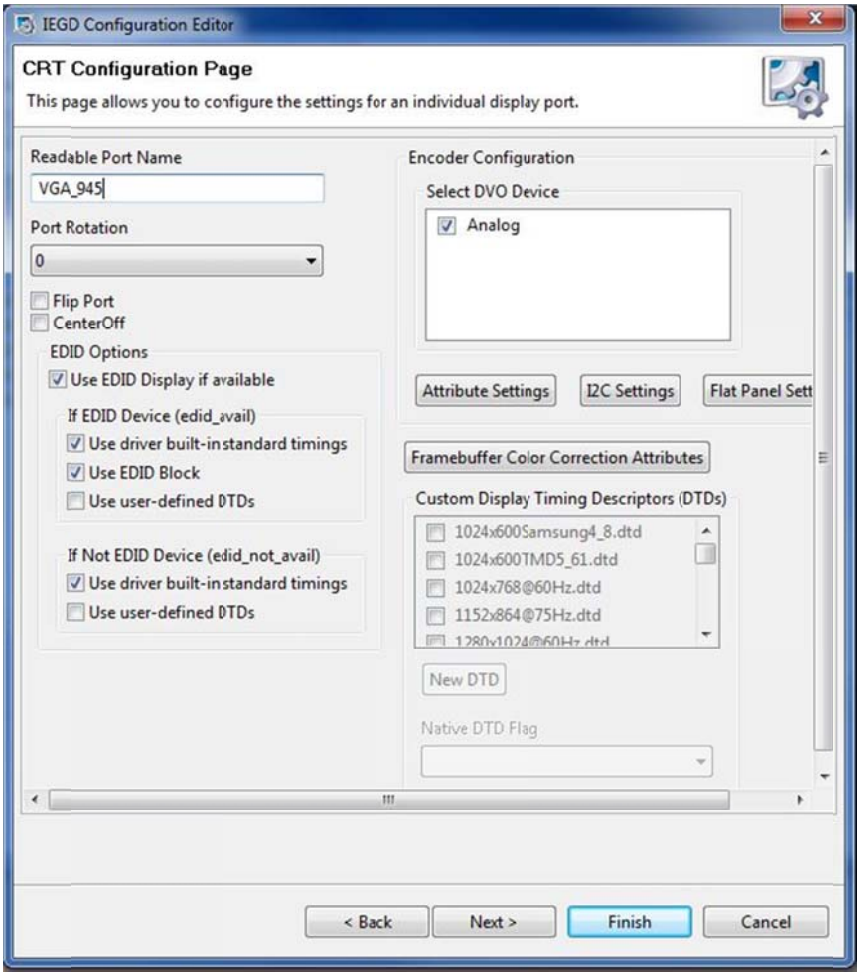

Figure 6 - VGA Configuration Page

7. The following are the settings for the VGA port:

   - Readable Port Name: **VGA_945**
   - Select DVO Device: **Analog**

8. Click **Finish**. The new configuration will be listed with the rest of the configurations.

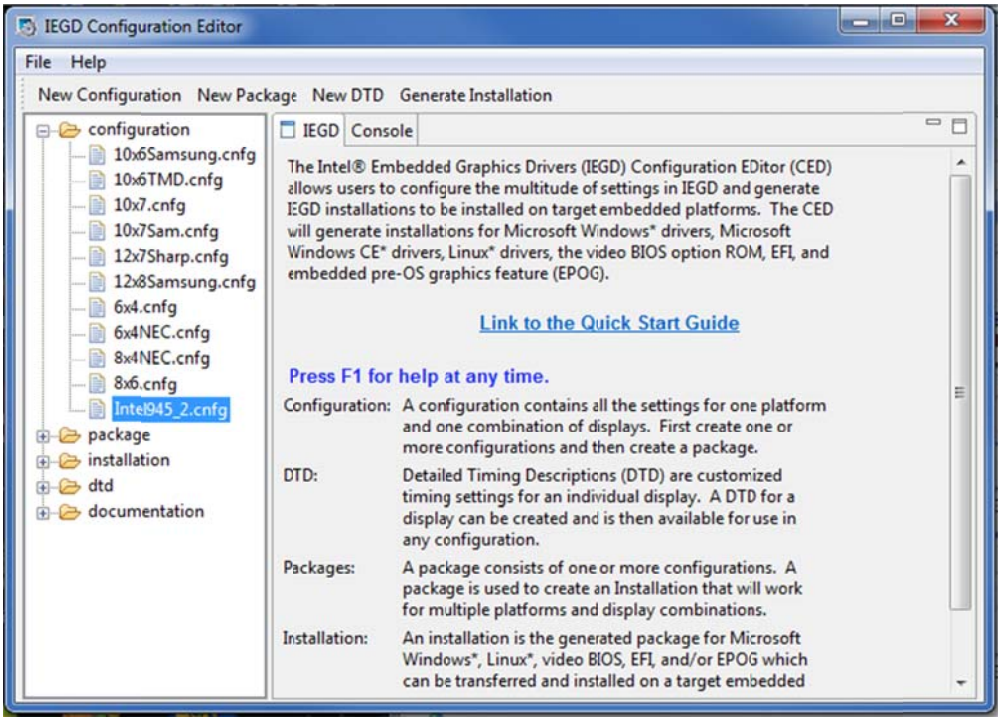

Figure 7 - New Configuration Added

9. The next step is to package the configuration with the operating systems you want to support. Click on the **New Package** from the menu. The IEGD package page will appear.

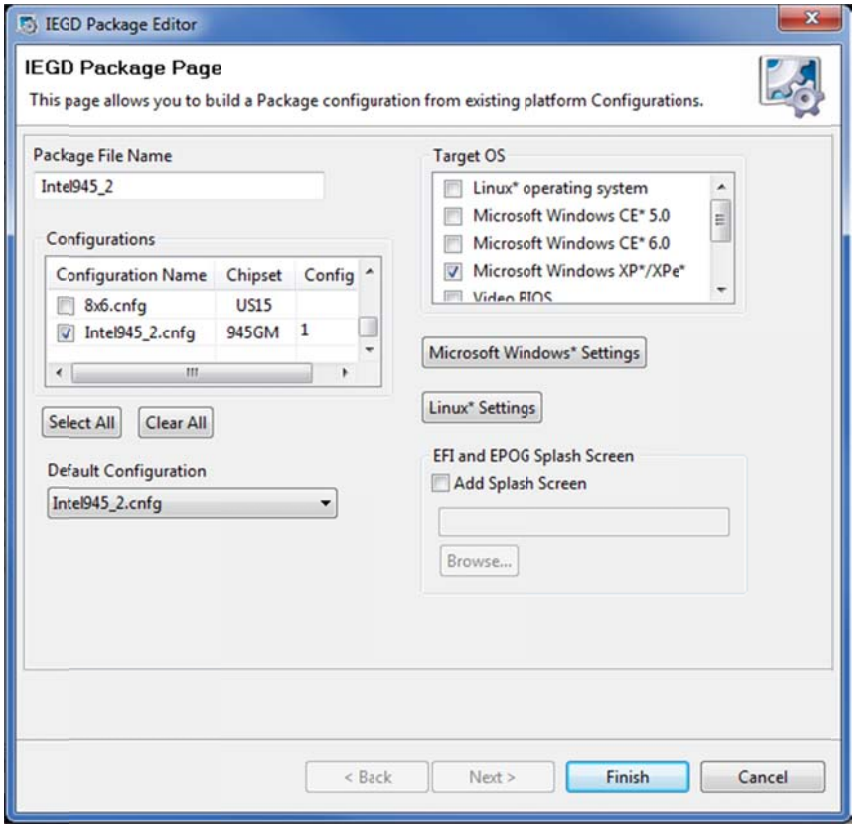

Figure 8 - Create a New Package

10. From the configuration list, select the I**ntel945_2.cnfg** configuration.
11. From the Target OS list, select **Windows XP/XPe**.
12. Click **Finish**.
13. The final step is to generate the installation package. Click on **Generate Installation** from the menu.

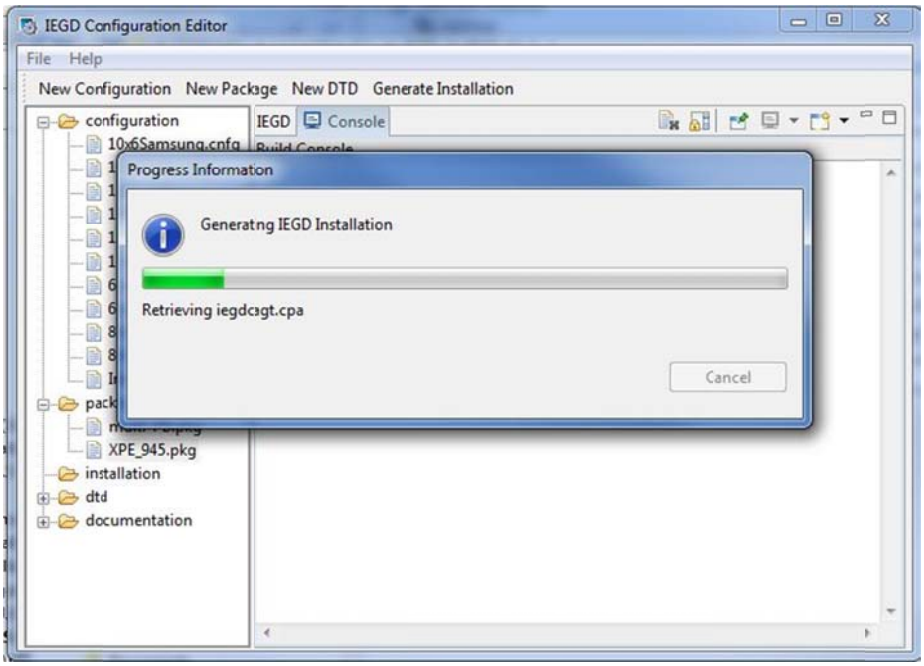

Figure 9 - Generating Installation Package

The final installation package will be placed in a zip file.

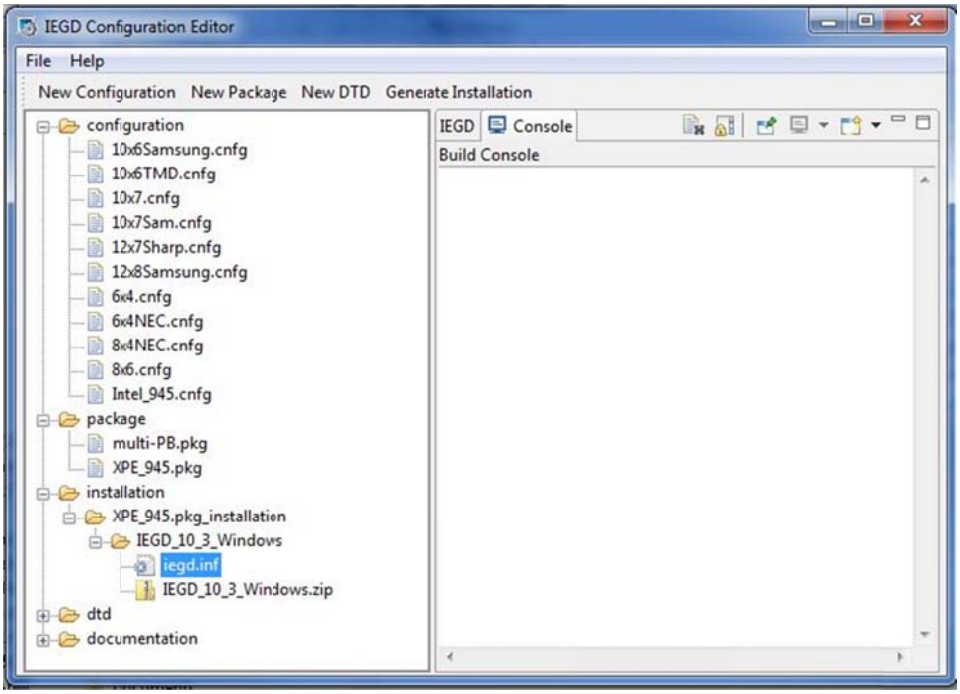

Figure 10 - New Installation Package Created

Notes on Creating the Driver SLD and adding Control Panel support

To create a component for the new driver created using the editor, you unzip the installer package that was generated. There are several folders in the extracted files. The Utilities and Driver folders are the most important for creating a component. If you want to just install the driver after the fact, you can buy running setup.exe found in the utilities folder. Creating a component will help remove this manual post-FBA step and better automate the building of the OS.

The basic steps to create the driver component are as follows:

1. Open Component Designer.
2. From the menu, File->Import. This will open the Import dialog. Make sure the file type is set to INF.
3. Go to the driver folder where the INF file is and open the file. Choose the defaults in the import wizard and let the INF be imported.
4. There will be several components that appear as the result. Delete all but one of the components. The only difference between these components is the name and PnP ID resource. Since the INF file is going to be in the image, Windows PnP manager has access to all PnP IDs and driver names. Technically, only one component is needed for one INF. The multiple component issue is an architecture design flaw with regards to Windows XP Embedded that was never fixed.
5. Rename the remaining component to **IEGD 10.x Driver**. The x can be whatever version you downloaded.
6. Save the SLD file to the same directory as the unzip package.
7. Create a repository and point the path to the driver's folder.
8. In the component, add the reference to the new repository. **Failure to do this step will result in an error when building the image in Target Designer.**
9. Create a package and in the group membership for the component and repository add the new package.
10. Save the SLD again.

The above are the basic steps to create a driver component. The driver will work just fine when added to the image, but you will notice that the Display Control panel doesn't include the advanced settings add-on for the Intel driver. The Utilities folder contains two DLL files that are required for the add-on: IEGDGUI.DLL and IEGDUI.DLL. There is also an extra utility that can be used independently of the Windows Display Control panel: IEGDGUI.EXE.

If you want to add extend control panel support, the SLD has to be modified.

1. Copy the IEGDGUI.DLL, IEGDUI.DLL, and the IEGDGUI.EXE files to the driver directory.
2. In the SLD, add the 3 new file resources and set the target paths to %11% (\windows\system32). The final listing should look like the following picture:

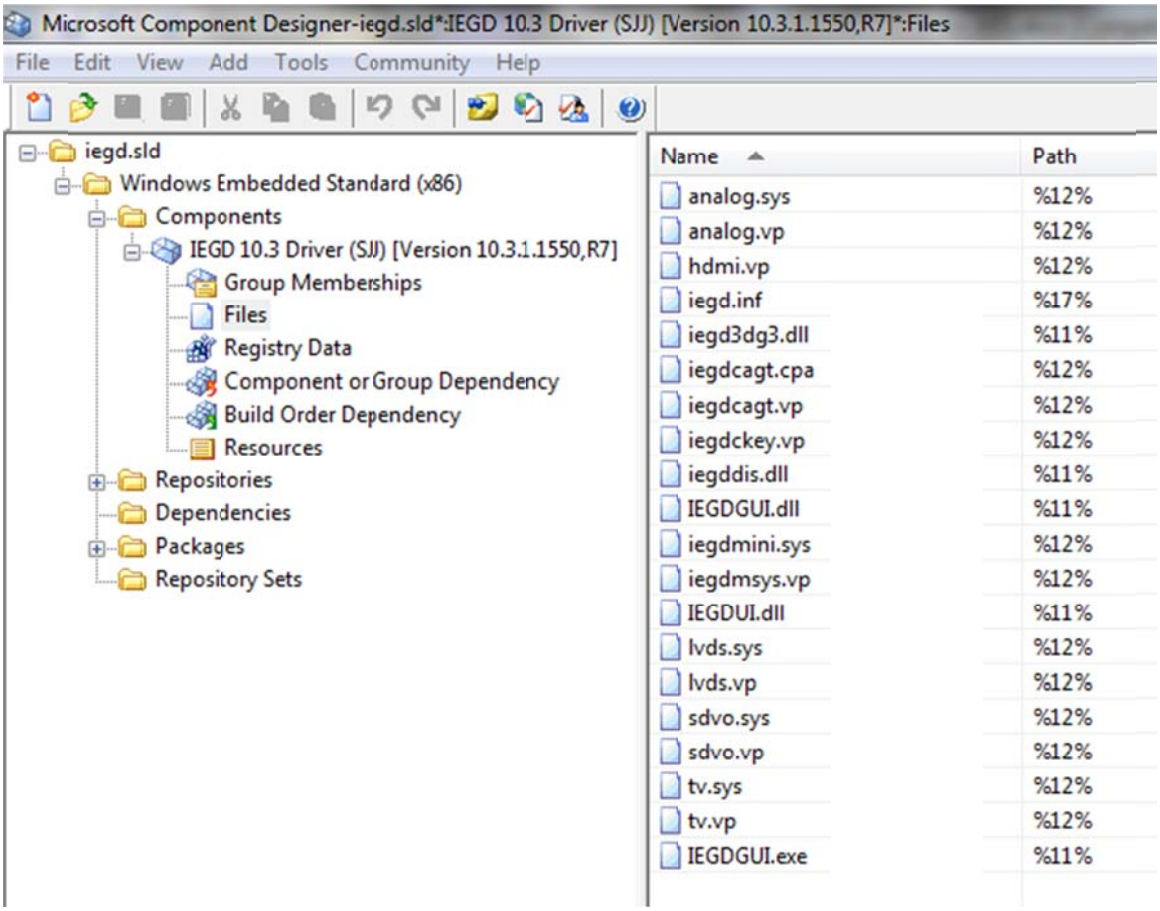

Figure 11 - Files Listed with all Driver and Control Panel Support Files

3. To properly display the information, two components are required for the proper codepage and fonts. To make sure the two components are in the image, in the Component or Group Dependency add the following components:

   - Codepage Application Compatibility
   - Fonts application Compatibility

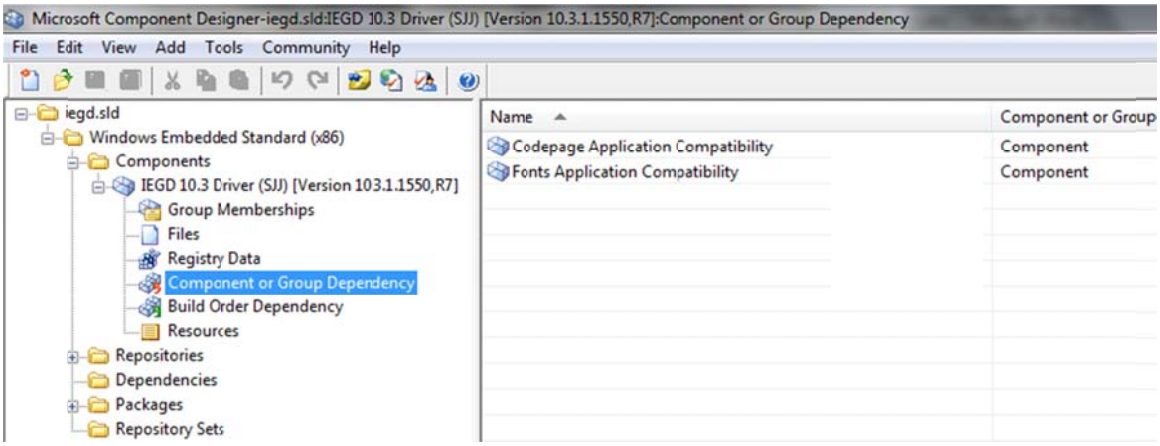

Figure 12 - Component Dependencies

4. Finally the IEGDGUI.DLL needs to be registered. Under Resources, add a new FBA DLL/COM Registration, and set the path to %11%\IEGDGUI.DLL.

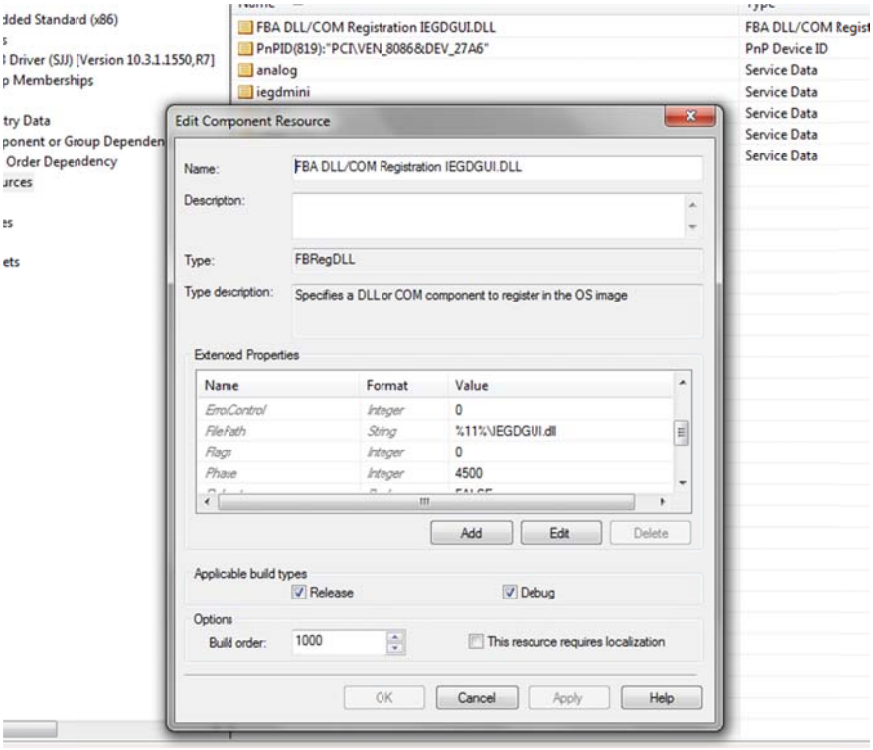

Figure 13 - IEGDGUI.DLL Registration

5. Save the SLD file.
6. Open Component Database Manager and import the new SLD. The component is ready for use in an image.

After the image has gone through FBA, you can see the advanced properties in the Display Control Panel.

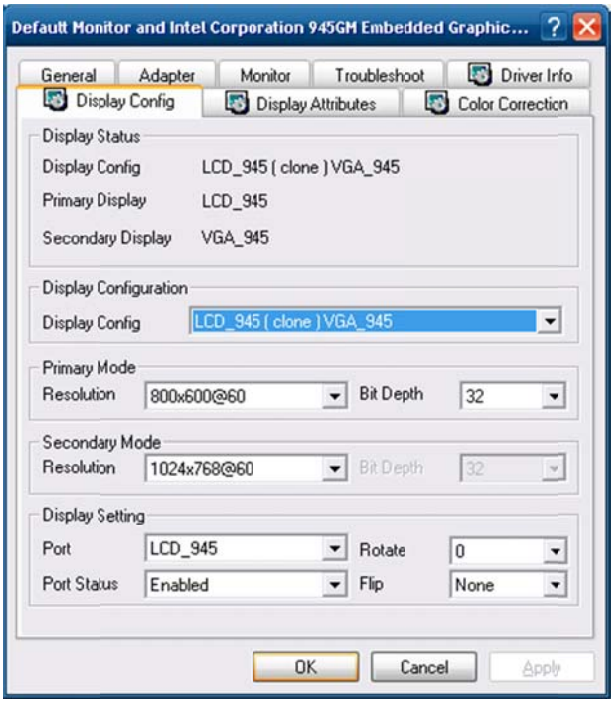

Figure 14 - Advanced Settings for the Intel Driver

You will notice that the port names match the name you created using the IEGD configuration editor. If you want to run the IEGDGUI.EXE application, you will notice that you only get the advanced properties.

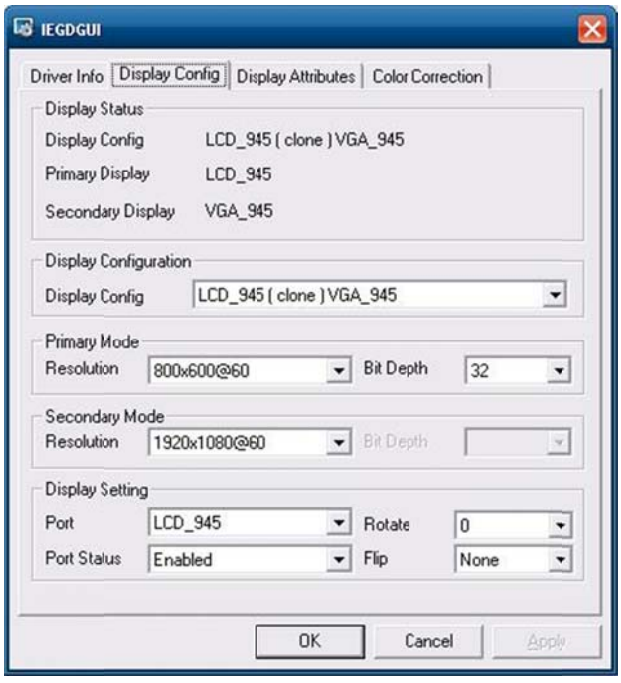

Figure 15 - IEGDGUI.EXE Application

You may get lucky and a generic video driver might fit your application, but the IEGD and EMGD provide a high level configuration solution to control what displays are supported. Best of all, the ability to create a component that doesn't require post-FBA manual operation makes it ideal for use with XPe\WES2009.

Annabooks is a wholly owned subsidiary of SJJ Embedded Micro Solutions.
Intel is a register trademark and Intel Atom is a trademark of Intel Corporation.
Windows is a registered trademark of Microsoft Corporation.