How-to Build: Windows Embedded Compact 7 Network Projector Adapter With an eBox-3350MX

Samuel Phung
Windows Embedded MVP
http://www.embeddedpc.net
# Table of Contents

Introduction .......................................................................................................................... 5  
Windows Network Projector Overview .............................................................................. 6  
eBox-3350MX Overview ....................................................................................................... 7  
Development Environment – Overview .............................................................................. 8  
  Windows Embedded Compact 7 ......................................................................................... 8  
  Platform Builder .................................................................................................................. 8  
  Windows Embedded Compact 7 Remote Tools .................................................................... 8  
Visual Studio 2008 Integrated Development Environment .............................................. 9  
Develop Compact 7 Devices: Typical Development Steps ................................................... 9  
Develop Compact 7 Applications with VS2008 ................................................................. 10  
Silverlight for Windows Embedded ..................................................................................... 10  
The Target Device ............................................................................................................... 10  
Common Terminology ........................................................................................................ 12  
Compact 7 Environment and Directory Variables ............................................................. 13  
OEM Environment Variables ............................................................................................ 13  
Development Environment – Connectivity ........................................................................ 14  
  Ethernet Connectivity ........................................................................................................ 14  
  Develop Compact 7 OS Runtime Image – OS Design ....................................................... 14  
  Develop Compact 7 Applications with Visual Studio 2008 ................................................ 14  
Serial Debug Connectivity .................................................................................................. 14  
Development Environment Setup with DHCP ................................................................. 15  
Using Wireless Access Point Router .................................................................................. 15  
Development Environment Setup with Static IP ............................................................... 16  
  Local Area Network without DHCP Service .................................................................... 16  
  Direct Connection using Cross-Over RJ-45 Ethernet Cable ............................................. 16  
  Static IP Addresses ............................................................................................................ 16  
Development Environment – Required Software & Installation ....................................... 17  
  Recommended Software Installation Sequence .............................................................. 17  
  Windows Embedded Compact 7 Installation .................................................................... 17  
Board-Support-Package Installation .................................................................................. 18  
Steps to Develop Compact 7 OS Design .......................................................................... 19  
  Visual Studio 2008 ............................................................................................................ 19  
  Windows Embedded Compact 7 OS Design Wizard ....................................................... 19  
OS Design Wizard – Board Support Package (BSP) .......................................................... 20
How to Build a Compact 7 Windows Network Projector Adapter with eBox-3350MX

Hardware: Debugging Considerations

Hardware: Real-Time Clock Considerations

Hardware: Data Bus Considerations

Hardware: Board Support Package Considerations

Hardware: Additional Components

Appendix-D: Windows Network Projector Display Resolution

Appendix-E: Connecting to a Windows Network Projector

To Connect to a Windows Network Projector

To Troubleshoot the Connection

Appendix-F: Windows Network Projector Registry Settings

Sample Network Projector Application Settings

Network Projector Service Settings
Introduction

This guide is written to show the steps to develop a Windows Network Projector adapter device, using an eBox-3350MX (eBox).

Using a proper hardware platform, with Windows Embedded Compact 7 (Compact 7) support, and the Compact 7 development environment, you can easily develop a Windows Network Projector solution and able to deliver proof-of-concept, working prototype within matter of weeks or even days.

Using an eBox as the target device, this guide provides information and works through multiple exercises to show the step to build a Windows Network Projector (WNP) adapter device.

The following topics are covered in this guide:

- Windows Network Projector overview.
- eBox-3350MX target device overview.
- Development environment – Overview.
- Development environment – Connectivity.
- Development environment – Required software and recommended installation sequences.
- Steps to develop Compact 7 OS design for a Windows Network Projector.
- Steps to configure and customize the OS design.
- Steps to build OS runtime image from the OS design.
- Steps to download the OS runtime image to the target device.
- Steps to deploy the OS runtime image to the target device’s local storage.
- Software Requirements and Considerations.
- Windows Embedded Compact 7 development environment.
- Develop a Windows Embedded Compact 7 OS design using the provided Windows Network Projector design template.
- Customize, configure and generate OS runtime image from the OS design.
- Download the OS runtime image to the target device (eBox-3350MX) for testing and debugging.
- Deploy the OS runtime image to the eBox-3350MX’s local storage to create a Windows Network Projector Adapter device.
- Connect a Windows 7-based PC to the Windows Network Projector adapter device.

After completing the exercises in this guide, you can transform the eBox-3350MX into a fully function WNP adapter device, which you can use to retrofit and existing projector with WNP feature.

The WNP feature provided in Compact 7 works with the supportive function built into the Windows 7 and Windows Vista operating system for the PC, which enable a Windows Vista or Windows 7 PC to display the desktop through the Windows Network Projector device, via a wired or wireless Network.

For more information about the Connect to a Network Projector feature, visit the following URL:

Windows Network Projector Overview

A Windows Network Projector (WNP) device has similar function as a typical projector that can accept video signal from a computer’s VGA interface.

In addition to the VGA interface, a WNP device is able to display the desktop from a Windows Vista or Windows 7 PC without the wire connecting to the VGA interface. Instead, the PC is able to stream the desktop display to the WNP device via a wired or wireless Network link.

The WNP OS design template, which includes all of the necessary software, is provided as part of the Compact 7 Platform Builder. Platform Builder is the development tool for Compact 7 that installs as a plug-in to the Visual Studio 2008 integrated development environment (IDE).

A WNP device built with Compact 7 is able to support the following usage scenarios:

- Microsoft PowerPoint presentations with simple animations and still image display.
- Displays to a single projector (one-to-one connection).
- Mirror or extended the PC’s desktop display.

The WNP device utilizes Remote Desktop Protocol (RDP) as the connectivity conduit to redirect the PC’s desktop display. It can support wired or wireless networks connections between the computer and the WNP device. It can support both ad hoc and infrastructure mode for wireless networks.

In addition to the function and feature provided as part of the WNP OS design template, you can modify the provided source code to add additional feature to enhance end-user experience.

Note:
The previous version of Windows Embedded Compact, Windows Embedded CE 6.0, also provides a Windows Network Projector design template.

For more information about Windows Network Projector, please refer to the appendix section toward the end of this guide.
eBox-3350MX Overview

The eBox-3350MX is a compact computing device designed for application where physical space and power consumption requirements are limited.

Engineer with an ultra-low power Vortex86MX System-On-Chip (SoC), the eBox-3350MX is built with the following features:

- 1.0 GHz Vortex86MX processor.
- 512 MB DDR2 system memory.
- XGI Z9s video support broad range of display resolution.
- 10/100 Mbps Ethernet.
- Audio.
- 3 USB 2.0 host interfaces.
- Boot from SD flash storage.
- Fan-less design.
- Power by +5 VDC via the Mini-USB interface.

Device drivers and board support packages (BSP) are available to support the following operating system:

- Windows Embedded Compact 7.
- Windows Embedded CE 6.0.
- Windows XP.
- Windows XP Embedded.

For more information about the eBox-3350MX, visit the following URL:

- [http://www.embeddedpc.net/eBox3350MX](http://www.embeddedpc.net/eBox3350MX)
Development Environment – Overview

The Windows Embedded Compact 7 Platform Builder (Platform Builder) development tool is a plug-in to the Visual Studio 2008 integrated development environment (VS2008 IDE). All Compact 7 development tasks are supported within the VS2008 IDE, a developer friendly, efficient and intuitive development environment with templates and wizard to help simplify complicate development tasks.

**Windows Embedded Compact 7**

Windows Embedded Compact 7 is a hard Real-time operating system with ability to handle 32,000 concurrent processes and 2GB memory footprint for each process. Compact 7 delivers reliable, secure performance in a small footprint package along with the latest networking, multimedia and communications technologies. Compact 7 provides broad range of device support with enhanced features, including robust file system, Web services for device, Silverlight for Windows Embedded, voice over IP, network gateway configurations, platform development tool enhancements, greater application compatibility with other Windows Embedded Compact based devices, Internet Explorer, Windows Media CODECs, Microsoft .NET Compact Framework, and a number of other newly supported protocols and services.

Combining large pool of production quality BSPs, device drivers, programming libraries and effective development tools, Compact 7 provides an ideal rapid application development environment to help create the next generation of smart, media rich, connected and service oriented devices. For more information about Compact 7, visit:


**Platform Builder**

Platform Builder is the development tool used to create Compact 7 OS design projects, build custom Compact 7 OS runtime images and develop Compact 7 device drivers. It also provides the remote tools to debug Compact 7 OS runtime image, device drivers and software components on a target device.

Platform Builder for Compact 7 is a plug-in to the VS2008 IDE, and requires the VS2008 IDE to function.

**Note:**

The previous version, Platform Builder for Windows Embedded CE 6.0, is a plug-in to the Visual Studio 2005 IDE.

For Windows CE 5.0 and earlier versions, Platform Builder is a standalone development tool.

**Windows Embedded Compact 7 Remote Tools**

Using remote tools provided as part of the Compact 7 development environment, developer is able to remotely debug Compact 7 OS runtime image on a target device.

The following remote tools are provided as part of the Compact 7 development environment:

- File Viewer.
- Kernel Tracker.
- Performance Monitor.
- Power Monitor.
How to Build a Compact 7 Windows Network Projector Adapter with eBox-3350MX

- Process Viewer.
- Profiler.
- Registry Editor.
- Resource Consumer.
- Resource Leak Detector.
- System Information.
- Timeline Viewer.
- Zoom.

**Visual Studio 2008 Integrated Development Environment**

Visual Studio 2008 is a popular developer friendly development tool to develop broad range of applications for different version of Microsoft Windows operating system. From the VS2008 IDE, developer is able to create applications for the following Windows operating system:

- Windows 7, Vista and XP.
- Windows Phone 7, Windows Mobile Smartphones and Windows Embedded Handheld devices.
- Windows Embedded CE 6.0 and Windows Embedded Compact 7 devices.
- More...

The VS2008 IDE provides a centralized development environment to develop broad range of applications, such as:

- Windows Form application.
- Win32, ATL and MFC project.
- DLL, ActiveX control, ASP.NET Web service, Windows service.
- SQL Server project, Console application, Class libraries.
- Windows Embedded Compact and Mobile Smart Device application.
- Windows Embedded Compact OS design.
- More...

**Develop Compact 7 Devices: Typical Development Steps**

After a proper hardware platform is selected, a typical Compact 7 device development project goes through the following development phases:

- Develop Compact 7 bootloader, device drivers and BSP for the selected hardware (some hardware vendors provide bootloader, device driver and BSP to support their hardware).
- Create and customize an OS design for the hardware.
- Build customized Compact 7 OS runtime image for the hardware.
- Deploy Compact 7 OS runtime image to the hardware for testing and debugging.
- Generate Compact 7 SDK from the OS design to support application development.
- Develop application for the Compact 7 device.
Application development can take place concurrently, as the hardware and OS design being fine-tuned.

After the satisfied Compact 7 OS runtime image and application are created, deploy the solution to the final hardware for distribution.

Develop Compact 7 Applications with VS2008
Visual Studio 2008 supports application development for Compact 7 devices. There are multiple options to develop Compact 7 applications from the VS2008 IDE:

- Develop Native code application with Visual C++.
- Develop Managed code application with Visual C#.
- Develop Managed code application with Visual Basic.

A Software Development Kit (SDK), generate from the OS design for the target device, is needed to support application development for the target device.

The Visual Studio IDE provides an efficient development environment, making it possible to download the OS runtime image and application from the development station to the target device for testing and debugging.

From the Visual Studio 2008 IDE with KITL and CoreCon connectivity, developer is able to download OS runtime image and application to a Compact 7 target device, launches the application, set breakpoint and step through the codes one-line-at-a-time as the application runs on the target device.

Silverlight for Windows Embedded
Silverlight is a user interface (UI) development framework designed to separate the graphical presentation from the programming logic. Silverlight enables graphic designer to design the graphical presentation for the application without the need to understanding the programming logic and provides a structured system for the application developer and graphic designer to work together, doing what they do best.

While the desktop and Windows Phone 7 Silverlight application’s code behind is based on managed code, Silverlight for Windows Embedded application’s code behind is based on native code, which provides an environment to develop small and efficient code to support embedded devices typically built with limited hardware resources.

Silverlight for Windows Embedded enables UI designers to develop stunning UI for the device without the need to understand complicated application code. At the same time, application developer can focus on their core expertise, develop the application logic and leverage UI graphic designer’s expertise to design user interface for the project.

Silverlight for Windows Embedded application for Compact 7 is developed using the combination of C++ and XAML code, using Visual Studio 2008 and Expression Blend 3.

The Target Device
Within the Compact 7 development environment, the term target device is referring to the hardware platform. For the exercises in this guide, the eBox-3350MX (eBox) is used as the target device.
The **ICOP_eBox3350MX_70C BSP** which includes the necessary hardware adaptation codes, device drivers and configuration files to support Compact 7 OS development for the eBox, is used for the exercises in this guide to develop the OS design project, as shown in Figure 1.

![Fig. 1 – ICOP_eBox3350MX_70C BSP for Compact 7](image)

A quality BSP greatly simplifies the tasks to create and configure an OS design to include the necessary components to generate an OS runtime image for the target device.
Common Terminology
To minimize the need to write long description and the need to use lengthy term repeatedly, it's a common practice for developer in different industries to use abbreviated key words. To the new comer, without knowing the terminologies used and what the abbreviated key words represent can cause confusion.

The following table contains abbreviation for some of the common key words used in the Compact 7 development environment and this guide:

<table>
<thead>
<tr>
<th>Abbreviation/ key word</th>
<th>Description and Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compact 7</td>
<td>Windows Embedded Compact 7</td>
</tr>
<tr>
<td>Platform Builder</td>
<td>Windows Embedded Compact 7 Platform Builder</td>
</tr>
<tr>
<td>VS2008</td>
<td>Visual Studio 2008</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>BSP</td>
<td>Board Support Package: A group of Windows Embedded Compact software components that includes the OEM adaptation layer code, device drivers and configuration files, needed to create the OS design project to generate OS runtime image for the targeted hardware.</td>
</tr>
<tr>
<td>OAL</td>
<td>OEM Adaptation Layer</td>
</tr>
<tr>
<td>Component</td>
<td>Component can be a device driver, BSP, programming library, application, utility, configuration settings, fonts, etc... The OS design is made up by a group of components.</td>
</tr>
<tr>
<td>Component Catalog</td>
<td>The component catalog lists all of the components provided by Platform Builder and installed 3rd party components. The component catalog provides the interface to add components to the OS design project and remove components from the project. This is also the interface to view which components are included to the OS design project.</td>
</tr>
<tr>
<td>OS design</td>
<td>OS design is a Platform Builder project, containing components that make up the OS runtime image. Windows Embedded Compact OS runtime image is generated from an OS design project.</td>
</tr>
<tr>
<td>Target Device</td>
<td>Refer to the device used in the Compact 7 development environment. When working on exercise using a VirtualPC, the VirtualPC is the target device. For the exercise in this guide, the eBox is the target device.</td>
</tr>
<tr>
<td>OS Runtime Image</td>
<td>The binary file generated from the OS design project to deploy to the target device.</td>
</tr>
<tr>
<td>Release Directory or Build Release Directory</td>
<td>Referring to the directory where all files related to the OS design projects are placed by the build tools prior to compiling the OS runtime image.</td>
</tr>
</tbody>
</table>
Compact 7 Environment and Directory Variables
Within the Compact 7 development environment, environment variables are used to configure the OS design to include, exclude associated components and control certain system behaviors. Directory variables representing the associated directory are used to help make the codes and script easier to understand and read. Here are some of these variables:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description - Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSP_DISPLAY_NOP</td>
<td>Configure the OS design to include the stub display driver (DDI_NOP.dll) to support headless device.</td>
</tr>
<tr>
<td>BSP_NOAUDIO</td>
<td>Configure the OS design to exclude all audio components from the build.</td>
</tr>
<tr>
<td>IMGRAM256</td>
<td>Configure the OS design to generate runtime image supporting 256MB of RAM</td>
</tr>
<tr>
<td>PRJ_ENABLE_FSREGHIVE</td>
<td>Configure the OS design to enable Hive-based registry support</td>
</tr>
<tr>
<td>_WINCEROOT</td>
<td>Represent the root directory for the Compact 7 install, default is: C:\WINCE700</td>
</tr>
<tr>
<td>_PLATFORMROOT</td>
<td>Represent the following directory: $(_WINCEROOT)\PLATFORM</td>
</tr>
<tr>
<td></td>
<td>Or C:\WINCE700\PLATFORM</td>
</tr>
</tbody>
</table>

The above table lists a small sampling of the variables used in the Compact 7 development environment. For more information about Compact 7 environment variables, visit the following URLs:

About Environment Variables:

BSP Environment Variables:

IMG Environment Variables:

PRJ Environment Variables:

Miscellaneous Environment Variables:

OEM Environment Variables
In addition to the environment variables used by Microsoft, third party developers can establish environment variables unique to their BSP to support the intended target device. The following environment variables are unique to the ICOP BSP for the eBox3350MX device:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description - Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSP_NIC_R6040</td>
<td>Enable support for R6040 Ethernet controller.</td>
</tr>
<tr>
<td>BSP_VMX_DISPLAY_xxxx</td>
<td>Multiple environment variables to configure display settings.</td>
</tr>
<tr>
<td>BSP_ENABLE_AgressiveFlushing</td>
<td>When enabled, it configures the OS design to perform aggressive registry flushing.</td>
</tr>
</tbody>
</table>
Development Environment – Connectivity

It’s important to establish a proper development environment and understand the different options available to establish connectivity between the development station and the target device, to download OS runtime image, and deploy application for testing and debugging. This section talks about connectivity between the development station and the target device, using an eBox as the target device.

Ethernet Connectivity
For the exercises in this guide, Ethernet is the primary connectivity used in the development environment with both the development station and target device connected to the same Local Area Network (LAN).

For the exercises in this guide, Ethernet connectivity is used to download the OS runtime image from the development station to the target device and deploy application to the target device for testing and debugging.

Develop Compact 7 OS Runtime Image – OS Design
Compact 7 OS design is one of the project type supported by the VS2008 IDE, to develop custom OS runtime Image.

As part of the OS design development process, connectivity between the development station and target device is needed to download OS runtime image to the target device. While it’s possible to use the serial port and other connectivity, Ethernet connection is efficient and provides the bandwidth to download Compact 7 OS runtime image quickly. Comparing to a serial connection, where the transfer rate is in the hundred kilo-byte (KB) range, Ethernet connection’s transfer rate is in the mega-byte (MB) range.

Develop Compact 7 Applications with Visual Studio 2008
The VS2008 IDE supports Compact 7 managed and native code applications development and provides the environment to deploy application from the development station to the target device.

As part of the application development exercises in this guide, with CoreCon connectivity established over an Ethernet connection, the VS2008 IDE provides the facility to deploy application to the target device, set breakpoint and enables the developer to step through the code, line by line, as the application executes on the target device.

Serial Debug Connectivity
Serial Debug connectivity is established by connecting a null RS-232 serial modem cable between a serial port on the target device and an available serial port on the development station. Serial connection is useful for debugging headless device and the device’s startup process during the bootloader phase where the Compact 7 OS runtime has not been fully launched.

A terminal emulation program, such as Hyper-Terminal or similar application, running on the development station, is used to capture serial debug messages from the target device.

Note:
The eBox-3350MX does not have a serial port.
Development Environment Setup with DHCP

One of the common development environments to support Compact 7 development is to attach both the development station and target device to the same LAN with DHCP service to provide IP addresses dynamically with a null RS-232 serial modem cable connected between one of the serial port on the target device and an available serial port on the development station.

Here is a typical setup to connect both the development station and target device to a LAN with DHCP service, as shown in Figure 2.

![Development environment with DHCP service provider](image)

Fig. 2 - Development environment with DHCP service provider

If the target device fails to establish connectivity with the development station as expected with this configuration, it may be caused by one of the following:

- The development station’s firewall software may be blocking the connection
- DHCP service for the target device needs to be enabled on your network. Some secured network may require the target device’s MAC address to be added to the authorized device list to be serviced by the DHCP server

Using Wireless Access Point Router

When using a wireless-access-point-router with multiple Ethernet ports, connecting both the development station and target device directly to the Ethernet port on the wireless-access-point-router may be problematic with certain model of access point, and prevent the development environment from functioning as expected.

**Note:**
The wireless-access-point-router device’s routing function filter and route network packets based on the packet’s associated origin, destination and other information. In some router, the routing algorithm may prevent some of the packets between the development station and target device from reaching its destination and cause problem.

To minimize connectivity problem, instead of connecting directly to the wireless-access-point-router device’s Ethernet ports, attach an Ethernet network hub to the wireless-access-point-router device, to access the DHCP service provided by the wireless-access-point-router, connect both the development station and target device to the Ethernet network hub. With both the development station and target device connecting to the same Ethernet hub, the network traffics between them are not filtered.
Development Environment Setup with Static IP

It's possible to establish a Compact 7 development environment with Static IP addresses. Here are two scenarios for setting up the development environment using static IP addresses:

**Local Area Network without DHCP Service**

The development station and target device are connected to the same LAN or Ethernet Hub, without DHCP service, as shown in Figure 3.

![Diagram of LAN without DHCP](image)

Fig. 3 - LAN without DHCP

**Direct Connection using Cross-Over RJ-45 Ethernet Cable**

Connectivity can be established by attaching the target device to the development station directly, using a cross-over RJ-45 Ethernet cable, as shown in Figure 4.

![Diagram of direct connection with cross-over RJ-45 Ethernet cable](image)

Fig. 4 - Direct connection with cross-over RJ-45 Ethernet cable

**Static IP Addresses**

Without DHCP service to assign IP addresses dynamically, the target device and development station must be configured with appropriate static IP addresses in order to establish connectivity. Both the development station and target device must be configured with static IP addresses within the same subnet.

Refer to appendix G for more information about using static IP addresses.
Development Environment – Required Software & Installation

The following software components are needed to work through the exercises in this guide:

- Visual Studio 2008
- Visual Studio 2008 service pack 1
- Windows Embedded Compact 7
- Visual Studio 2008 update for Windows Embedded Compact 7
- ICOP_eBox3350MX_70C BSP

Recommended Software Installation Sequence

It’s important to install the software in their proper sequences. Here is the recommended software installation sequence:

- Visual Studio 2008
- Visual Studio 2008 service pack 1
  Visual Studio 2008 service pack 1 is available from the following URL:
- Windows Embedded Compact 7
  A 180 days evaluation version of Compact 7 software is available for download from Microsoft, at the following URL:
- Visual Studio 2008 update for Windows Embedded Compact 7
  VS2008 update for Compact 7 is available for download from Microsoft, at the following URL:
- ICOP_eBox3350MX_70C_BSP.msi
  The eBox-3350MX BSP for Compact 7 is available for download from the following URL:
  http://www.embeddedpc.net/eBox3350MX/

Windows Embedded Compact 7 Installation

Since the Compact 7 development tool, Platform Builder, is a plug-in to the VS2008 IDE, VS2008 must be installed to the develop workstation prior to installing the Compact 7 software.

When installing the Compact 7 software, during the installation options selection, select the Custom install option, as shown in figure 5.
During the supported processor selection step, include support for the x86 Architecture, as shown in Figure 6.

Note: Selecting all processors will take up more than 50 GB of storage to install the software.

**Board-Support-Package Installation**

VS2008 and Compact 7 must be installed prior to installing the Board-Support-Package (BSP). Download and install the ICOP_eBox3350MX_70C BSP. After installation, this BSP shows up on the component catalog as “ICOP_eBox3350MX/70C : x86”, under the “\Third Party\BSP” folder.
Steps to Develop Compact 7 OS Design

In this section, with help from the *OS Design wizard*, you will work through the steps to create a new OS design project.

**Visual Studio 2008**

The VS2008 IDE provides support to create different type of projects, such as Windows Application, Console Application, Class Library, smart device application, Windows Services, Web Control, etc... When starting a new project with VS2008, the VS2008 IDE provides different wizards and templates to help create the initial workspace for the project. Platform Builder is one of the available project types, to develop Compact 7 OS design project.

To begin, launch VS2008. From the VS2008 menu, select **File → New → Project** to bring up the new project screen, as shown in Figure 7.

![Fig. 7 - VS2008 New Project – Platform Builder 2008 (OS design)](image)

- From the *New Project* screen’s left pane, click to highlight the **Platform Builder** option.
- From the right pane, click to highlight the **OS Design** option.
- Enter a project name. For this guide, let’s use **3350Projector** as the name for the project.
- Make sure the **Create directory for solution** check box is checked.
- Click **OK** to continue.

**Windows Embedded Compact 7 OS Design Wizard**

When a new Platform Builder OS design project is selected, the *OS Design Wizard* launches, as shown in Figure 8, and guides you through the process to configure the initial OS design project using templates available as part of the VS2008 IDE.
Fig. 8  -  OS Design Wizard

- Click **Next** to continue and bring up the *Create an OS Design - Board Support Packages* selection screen.

OS Design Wizard – Board Support Package (BSP)

In the BSP selection step, the *OS Design Wizard* provides the options to select one or more BSP for the new project from the list of available BSPs. All of the installed BSPs, including BSPs from Microsoft and third party companies are listed on the *Create an OS Design - Board Support Packages* screen, as shown in Figure 9.

Fig. 9  -  OS design Wizard – Select BSP

- From the *Select one or more BSPs* pane, select the **ICOP_eBox3350MX_70C: x86** BSP.
- Click **Next** to continue and bring up the *Create an OS Design - Design Templates* screen.
OS Design Wizard – Design Templates
In the design template selection step, the OS Design Wizard provides multiple design templates to choose from, as shown in Figure 10.

Fig. 10 - OS Design Wizard – Design Templates

- Expand the Enterprise Device node and select the Network Projector template. Click Next to continue and bring up the Create an OS Design - Application & Media screen.

OS Design Wizard – Applications & Media
In the applications & media selection step, the OS Design Wizard provides the options to select error reporting component, as shown in Figure 11.

Fig. 11 - OS Design Wizard – Applications & Media

For the exercise in this guide, let’s ignore the error reporting subject. Click Next to continue and bring up the Create an OS Design - Networking & Communication screen.
OS Design Wizard – Networking & Communications

In the networking & communication selection step, the OS Design Wizard provides the options to select communication, networking and security components, as shown in Figure 12.

![OS Design Wizard - Networking & Communications](image)

Fig. 12 - OS Design Wizard – Networking & Communications

- For the networking & communication selection step, keep the default selection. Click **Next** to continue and bring up the Create an OS Design - OS Design Project Wizard Complete screen, as shown in Figure 13.

At this point, the OS Design Wizard included the necessary OS components needed to configure the initial OS design project.

![OS Design Project Wizard Complete](image)

Fig. 13 - OS Design Wizard – Completed

- Click **Finish** to complete the OS Design Wizard step.
Security Warning - Catalog Item Notification

At the completion of OS Design Wizard step, the Catalog Item Notification screen is raised, with security warning, to warn one or more of the components included in the OS design may pose security risk, as shown in Figure 14.

Fig. 14 - Catalog Item Notification – Security Warning

- Click Acknowledge to acknowledge the warning and close the warning screen.

At the completion of the OS Design Wizard step, the initial project workspace for the 3350Projector OS design is created in the following directory:

- C:\WINCE700\OSDesigns\3350Projector\
Steps to Configure and Customize the OS Design

At this point, the initial workspace for the 3350Projector OS design project is created using the Network Projector design template along with the ICOP_eBox3350MX_70C BSP and components selected during the OS Design Wizard steps.

The following project folder and sub-folders are created for the OS design, under the main Compact 7 OS design project directory:

- C:\WINCE700\OSDesigns\3350Projector\ This is the folder for the 3350Projector solution. VS2008 supports different project types. A solution provides a centralized work space to keep different project types supporting the same solution in one location.
  For example, the 3350Projector solution may include the “3350Projector OS design”, “Visual Basic managed code application”, “Visual C# managed code application” and “Visual C++ native code application”.
- C:\WINCE700\OSDesigns\3350Projector\3350Projector\ This is the folder for the 3350Projector OS design project.

The VS2008 IDE should look similar to the screen as shown in Figure 15.

Fig. 15 - VS2008 IDE after completing the OS Design Wizard steps
Customize the OS Design – Add Additional Components

The OS design can be further customized with the following:

- Add component(s) to the OS design.
- Remove component(s) from the OS design.
- Add application and library as subproject to the OS design.
- Modify system configuration and registry files to customize system behavior.

With the **3350Projector** OS design project active, from the VS2008 menu, select **View → Other Windows → Catalog Items View** (you can also click on the Catalog Items View tab) to bring up the Catalog Item View window, as shown in Figure 16.

![Catalog Items View](image)

**Fig. 16 - Catalog Items View (Component Catalog)**

The **Catalog Item View** window on the VS2008 IDE contains all of the Compact 7 components, including application, library, driver, utility and 3rd party components installed to the development station which can be added to the OS design project.

From the **Catalog Items View** window, expand the component nodes and review the components selected during **OS Design Wizard** steps:

- Component with a green check mark to the left indicates it was selected during the **OS Design Wizard** steps, or manually added after the **OS Design Wizard** steps.
- Component with a solid green square to the left indicates the component is included to the OS design as the result of being a dependency to another selected component.
- Component with a red cross to the left indicates the component is selected and is dependent on one or more other components currently not included to the OS design. (Component with a red cross to the left will not be included to the final image.)

Work through the following steps to customize the **3350Projector** OS design project:

- From the **Catalog Items View** window, expand the **\Third Party\BSP** node.
- Expand all nodes under the **ICOP_eBox3350MX_70C: X86** node, as shown in Figure 17.
Verify the following BSP components are selected and included to the OS design:

- **ATAPI (IDE) Storage driver**
  
  **Note:**
  The BSP’s ATAPI (IDE) Storage component set the SYSGEN_ATAPI variable to include the ATAPI storage driver, and the SYSGEN_FATFS variable to include FAT file system support.

- **Hive-based registry support**
  
  **Note:**
  The BSP’s Hive-based registry component is needed to save registry settings to non-volatile flash storage between power reset. The Hive-based registry component is a sub-component under the ATAPI (IDE) Storage driver component.

- **Enable Aggressive Flushing**
  
  **Note:**
  This is a sub-component to the Hive-based registry component. When enabled, it configures the OS runtime image to automatically save changes to the registry.

- **Vortex86MX display driver**
  
  **Note:**
  Device driver to support the built in Vortex86MX display controller.

- **VGA-05 1024x768x16 @ 60Hz**
  
  **Note:**
  This is a sub-component to the Vortex86MX display driver. 1024x768 16-bit display mode is selected for this exercise. You can select a different display mode to support the projector you are working with.
How to Build a Windows Network Projector Adapter

- **R6040 Ethernet driver**  
  **Note:**  
  Device driver to support the R6040 Ethernet controller.

- **PS/2 Keyboard & Mouse driver**  
  **Note:**  
  This BSP component set the BSP_KEYBD_8042 environment variable to include the PS/2 Keyboard and Mouse driver to the OS runtime image.

- **512MB RAM**  
  **Note:**  
  There are different models of eBox, built with 256MB or 512MB of system memory. This component set the IMGDRAM512 environment variable and configure the OS runtime image to use the 512MB of available system memory. Improper system memory configuration can cause the Compact 7 OS runtime image not able to complete the boot up process and reset the device.

- **USB Keyboard & Mouse (HID) class driver**  
  **Note:**  
  This component set the SYSGEN_USB_HID_CLIENTS environment variable to include the USB Keyboard and Mouse HID class driver to the OS runtime image.

### Customize the OS Design – Locate Component by Search

The VS2008 IDE provides the search function to locate Compact 7 component from the catalog by searching the catalog using key-word or partial key-word associated with the component. In this section, you will use the search function to locate a component from the catalog and add the component to the OS design.

From the **Catalog Item View** tab, enter **Network Projector** in the search text box on the top right, and click the **green arrow** to the right of the search text box to locate the **Telnet Server** component, as shown in Figure 18.

![Catalog Items View](image)

**Fig. 18 - Locate component using the search function**

The search engine locates and highlights the **Windows Network Projector** component, which is already included to the OS design during the OS design wizard steps.
Customize the OS Design – Configuration Manager

Using the configuration manager, the OS design can be configured to generate an OS runtime image in checked, debug or release mode. An OS image built in release mode is optimized for distribution, does not contain ASSERT and not able to output DEBUGMSG. An image built in debug mode is not optimized and include additional resource to support debugging. However, a debug mode image, without optimization, can be difficult to debug certain type of timing and performance related problems. An image built in Checked mode (Checked mode is new to Compact 7), a hybrid between the debug and release modes, is built with compiler optimization, contains ASSERT, DEBUGMSG and RETAILMSG to provide the necessary resources needed for debugging.

A debug mode image will provide more detailed debug messages when the OS image loads and executes applications and modules. The size of an OS image built in debug mode is generally about 50% larger than an OS image built in release mode, from the same OS design project. An OS image built in release mode, with KITL enabled, provides sufficient debug information to support most of the general application development environment. For the exercise in this guide, you will configure the OS design to generate an OS runtime image in release mode.

- From the VS2008 menu, select Build → Configuration Manager to bring up the Configuration Manager screen, as shown in Figure 19.

![Configuration Manager](image)

Fig. 19 - Configuration Manager

- From the Active solution configuration selection options, select ICOP_eBox3350MX_70C x86 Release and click Close to configure the OS design to generate a release mode image.

By default, the 3350Projector OS design is configured to generate runtime image in release mode. The exercise in this section demonstrates the available options to configure the OS design to generate runtime image in checked, debug and release modes.

Customize the OS Design – Build Options

The OS design can be further customized by editing the project’s build options.

From the VS2008 menu, select Project → 3350Projector Properties to bring up the 3350Projector Property Pages screen, as shown in Figure 20.
From the left pane, expand the *Configuration Properties* node and select *Build Options*.

From the right pane, change the *Enable KITL* build option to *No* as shown in Figure 21.

**Note:**
When deploying a Compact 7 OS runtime image, with KITL enabled, to the target device’s local flash storage, the system will search for an unavailable KITL connection during start up and will not be able to complete the boot process.

Click **Apply** and **OK** to close the *3350Projector Property Pages* screen.

**Customize the OS Design – The Registry**

The registry plays a key role in controlling how the Compact 7 OS runtime behave, loads driver, application and more. Improper registry entries can cause serious problem and can cause the system failing to complete the boot process.

For the exercise in this guide, the required registry entries to launch the device drivers and application components included to the OS design project are configured by the OS design wizard and the eBox-3350MX BSP.
For the exercise in this section, let’s work through the following steps to add registry entries to configure the device name and Ad hoc SSID:

1. Click on the Solution Explorer tap to bring focus to the Solution Explorer.

2. Expand the Parameter Files folder and double click on OSDesign.reg to open this registry file in the code editor as shown in Figure 22.

![Fig. 22 - OSDesign.reg registry file](image1)

3. From the code editor’s lower left, click on Source to view the registry entries in source code format, as shown in Figure 23.

![Fig. 23 - OSDesign.reg registry file in source code format](image2)

4. Add the following registry entries to the OSDesign.reg file.

```
[HKEY_LOCAL_MACHINE\Software\Microsoft\PictorService]
"ProjectorName"="3350MX Projector"
"AdhocSSID"="3350MXProjector"
```

**Other Compact 7 Components**

In addition to the components selected during the OS design Wizard step and the components automatically included in the OS design by the template, additional components from the catalog can be added to the OS design to provide additional function and features.

For example, the File Server component can be added to provide files and folders sharing over a network connection. The FTP Server component can be added to provide remote file upload and download services. The RAS Server/PPTP Server (Incoming) component can be added to provide inbound dialup network connection via the serial port.
The following table lists the eBox’s I/O peripherals and their associated Compact 7 device drivers and support components:

<table>
<thead>
<tr>
<th>eBox Peripherals</th>
<th>Compact 7 Drivers &amp; Support Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGA display</td>
<td>Vortex86MX display driver <em>(Driver included with BSP)</em></td>
</tr>
<tr>
<td>Audio</td>
<td>HD Audio driver <em>(Driver included with BSP)</em></td>
</tr>
<tr>
<td>Network</td>
<td>R6040 Ethernet driver <em>(Driver included with BSP)</em></td>
</tr>
<tr>
<td>USB Interfaces</td>
<td>Provided as part of the BSP</td>
</tr>
<tr>
<td>SD**¹</td>
<td>ATAPI <em>(Driver available from Platform Builder’s component catalog)</em></td>
</tr>
</tbody>
</table>

**¹ The SD slot on eBox is link to the IDE interface, and does NOT support hot-swap. When a SD storage card is inserted into the slot prior to power on, the system will detect and recognize the SD storage card as an IDE storage device. When booting to Compact 7, the SD storage card will shows up as “Hard Disk”.

The eBox used to develop the exercise for this guide is configured to boot from the SD storage card. The SD slot on the eBox is link to the IDE interface to support SD storage card. The SD slot on the eBox does not support I/O module. SD storage card inserted to the SD slot on the eBox behaves just like a typical IDE hard disk and requires the ATAPI driver and FAT file system components to function.

To learn more about other Compact 7 components on the Platform Builder component catalog, refer to the Compact 7 help document on the development station, or the following MSDN URL:

Windows Embedded Compact Documentation

Steps to Build and Generate OS Runtime Image

In the previous sections, you created an OS design project, configured and customized the OS design. In this section, you will work through the steps to build and generate the Windows Network Projector OS runtime image for the eBox.

The Build Process – Starting

From the VS2008 menu, with the 3350Projector OS design project active, select Build → Advanced Build Commands → Sysgen (blddemo -q) to build and generate an OS runtime image from the OS design.

Depending on the development station’s processor speed and available system resources, the build process may take anywhere from 10 to well over 30 minutes.

During the build process, the output tab on the VS2008 IDE displays compilation activities, as shown in Figure 24.

![VS2008 IDE showing the OS design being built.](image)

The Build Process – Completed

When the build process is completed, the VS2008 IDE should look similar to the screen, as shown in Figure 25.
The Output tab on the VS2008 IDE shows the result of the build process, with “0” failed task.

**Note:**
When the build process ended with 1 or more error, the build process failed and will not generate a runtime image. When the build process ended with warnings and without any error, the build process is completed with a runtime image file generated.

The number of warning may vary depending on the selected components and installed QFE. When working with a known good BSP, the warnings are generally non critical and do not impact system function. However, as part of good engineering practice, it’s good to go through the warning messages to identify potential problem.

With a successful build, a Compact 7 OS runtime image file, *NK.BIN*, is generated in the following build release directories:

- For OS design configured to generate an image in checked mode:
  \WINCE700\OSDesigns\3350Projector\3350Projector\RelDir\ICOP\_eBox3350MX_70C_x86_Checked

- For OS design configured to generate an image in debug mode:
  \WINCE700\OSDesigns\3350Projector\3350Projector\RelDir\ICOP\_eBox3350MX_70C_x86_Debug

- For OS design configured to generate an image in release mode:
  \WINCE700\OSDesigns\3350Projector\3350Projector\RelDir\ICOP\_eBox3350MX_70C_x86_Release

After the build process is completed, Platform Builder generates the following log files for the last build process:

- C:\WINCE700\Build.log (This file list all of the build steps in detail.)
- C:\WINCE700\Build.wrn (This file list all of the warnings during the build process.)
- C:\WINCE700\Build.err (This file list all the errors during the build process. This file is not present for a successful build.)
Steps to Download the OS Runtime to Target Device

To download the runtime image to the target device, you need to establish connectivity between the VS2008 development station and target device.

In this section, you will work through the steps to establish connectivity and download the OS runtime image created in the earlier steps to the target device.

For target device built with other processor and I/O peripherals different from the eBox, it may require different transport mechanism to establish the connectivity. The eBox is built with an x86 processor (Vortex86MX) and will use the integrated Ethernet interface to establish connectivity to the VS2008 development station.

To work through the exercises in this guide, both the development station and target device must be connected to the same LAN, using one of the following connectivity options:

1. Both the development station and target device are connected to the same LAN with DHCP service to provide IP addresses dynamically.
2. Both the development station and target device are connected to the same LAN without DHCP service and need to configure static IP addresses.
3. The target device is connected to the development station directly using a cross-over Ethernet cable and need to configure static IP addresses.

Note: Refer to appendix A and B for more information about connectivity for downloading OS runtime image from the development station to target device.

The exercise in this section is created with both the development station and target device connected to the same LAN with DHCP service to provide IP addresses dynamically.

In order to establish connectivity to the Compact 7 development station to download OS runtime image, the target device needs to launch an appropriate bootloader.

Refer to Appendix-A for more information about configuring the SD flash storage with BIOSLoader and Ethernet bootloader using DiskPrep.

Create Connectivity Profile for a Target Device

To establish connectivity and download Compact 7 OS runtime image from the development station to the target device, you need to configure a target device connectivity profile and associate the profile to the target device in use.

In this section you will work through the steps to create a target device connectivity profile and associate this profile with the eBox, as the target device for the exercise in this guide.

Note: The exercise in this section is applicable to development environment using dynamic IP addresses with DHCP, and development environment using static IP addresses.

Continue from the previous exercise, with the 3350Projector OS design project active, from the VS2008 menu, select Target → Connectivity Options to bring up the Target Device Connectivity Options screen, as shown in Figure 26.
Work through the following steps to create a connectivity profile for the target device (eBox):

- From the **Target Device Connectivity Options** screen, click **Add Device** to bring up the screen to add a new target device profile as shown in Figure 27.

Enter **MyTargetDevice** as the new target device name

Click **Add** to continue

The **Target Device Connectivity Options** screen displays the new device profile, **MyTargetDevice**, as shown in Figure 28.
Associate eBox to MyTargetDevice Connectivity Profile
In this section, you will work through the steps to associate the eBox to the MyTargetDevice connectivity profile, created in the previous section.

From the Target Device Connectivity Options screen (continuing from previous section), clicks on the top most Settings button, to bring up the Ethernet Download Settings screen, as shown in Figure 29.
The *Ethernet Download Settings* screen is waiting for BOOTME messages from the target device’s Ethernet bootloader (*EBOOT.BIN*).

Work through the following steps to associate the eBox to this device profile:

- **Apply power to the target device.**
- **As** the target device power up and display the Windows Embedded Compact 7 splash screen, press “1” to launch the Ethernet bootloader, *EBOOT.BIN*.
- **As** the *EBOOT.BIN* Ethernet bootloader is launched, it broadcasts a series of BOOTME messages to the attached LAN via UDP.
- **As** the *Ethernet Download Settings* screen detected the BOOTME messages from the target device, it displays the device ID from the target device, as shown in Figure 30.

![Ethernet Download Settings](image)

**Fig. 30** - *Ethernet Download Settings*

- **Click and highlight** the device ID listed in the *Active target devices* windows to associate the device ID to the profile.
- **Click** *Apply* to continue.

**Note:**

In the environment where there are multiple target devices connected to the same network segment booting up and send bootme requests at about the same time, multiple device IDs will be listed in the Active Devices windows. To identify the target device you are working with, make sure it’s the only target device booting and sending the bootme request.

After the device ID is detected and accepted in the *Active target devices* step, the same device ID is now shown on the *Target Device Connectivity Options* screen, as shown in Figure 31.
Fig. 31 - Ethernet Download Settings

- Click **Apply** and **Close** to save the settings and close the *Target Device Connectivity Options* screen.

**Download OS Runtime Image to Target Device**

With the *MyTargetDevice* connectivity profile created and the eBox target device associated with this profile, you are ready to download the OS runtime image from the Development station to the target device.

Work through the following steps to download the OS runtime image to the target device:

- From the VS2008 IDE, select and use *MyTargetDevice* device profile as shown in Figure 32.

Fig. 32 - select MyTargetDevice from VS2008 IDE

- From the VS2008 menu, select **Target → Attach Device** to initiate the download process and bring up the *Device Status* screen, as shown in Figure 33.
Power on (or reset power) the target device.

As the Compact 7 splash screen is showing, press “1” to launch the Ethernet bootloader, EBOOT.BIN, before the 10 seconds delay is timed out.

As the EBOOT.BIN launches, it sends a series of BOOTME messages to the development station to trigger the OS runtime image download process.

As the image download takes place, the Device Status screen displays download activities, as shown in Figure 34.

Note: Once launched, the Ethernet bootloader, EBOOT.BIN, will continue to broadcast BOOTME messages to the development station and timed out after about 3 minutes. If you failed to initiate the OS runtime image before the timed out period, you need to re-initiate the download process again.

After the image download process is completed, the Compact 7 OS runtime image downloaded to the target device will launch. Be patient... It takes a few moments for the image to boot up.

When the Windows Embedded Compact 7 screen is displayed on the target device, it’s an indication the boot process is completed, as shown in Figure 35.
Deploy Compact 7 OS Runtime Image to Target Device

In the previous step, when the Compact 7 OS image is downloaded and launched on the target device, the image is placed directly to the device’s memory (RAM) and is not saved to the device’s local storage. After going through power reset, the downloaded image will be lost. To deploy a Compact 7 OS runtime image to the target device’s local storage, the OS runtime image file, NK.BIN, needs to be copied to the local storage and configured with an appropriate bootloader.

The NK.BIN OS runtime image file, in release mode, is generated in the following directory:

```
C:\WINCE700\OSDesigns\3350Projector\3350Projector\RelDir\ICOP_eBox3350MX_70C_x86_Release
```

**Note:**

- Release mode image built with KITL enabled and debug image is not intended for deployment to the target device’s local storage.
- When deploying a release mode OS runtime image, generated with KITL enabled, to the target device’s local storage, the image attempt to establish connectivity to an unavailable KITL connection and will not be able to complete the boot process.

There are multiple methods to deploy the NK.bin OS runtime image file to the target device’s local storage.

To deploy a Compact 7 OS runtime image to the target device’s local storage, the OS design must be configured with the following:

- Generate OS runtime image in release mode
- Disable the KITL build option and other build option configured to support debugging

Refer to Appendix-A for more information about deploying a Compact 7 OS runtime image to the target device’s local storage using DiskPrep, a Windows Embedded Compact power toy.

The DiskPrep power toy is available for download from the following URL:

```
http://code.msdn.microsoft.com/DiskPrep
```
Appendix-A: Preparing SD Flash Storage to Launch Compact 7 OS and Ethernet Bootloader

In order to launch Compact 7 OS runtime image or Ethernet bootloader, the SD flash storage must be configured with an appropriate bootloader. For the exercise in this guide, the SD storage is configured with a BIOSLoader.

To prepare the SD flash storage with BIOSLoader to launch an Ethernet bootloader to establish connectivity with the development station, to download OS runtime image, you need the following:

- Ethernet bootloader for the eBox, an EBOOT.BIN file. It’s available for download from the following URL:
  http://www.embeddedpc.net/eBox3350MX/
- DiskPrep utility from the following URL:
  http://code.msdn.microsoft.com/DiskPrep

Configuring SD Flash Storage using DiskPrep

The DiskPrep utility can be used to configure different type of storage with BIOSLoader to launch a Compact 7 OS runtime image or an Ethernet bootloader with customizable startup splash screen.

To configure the SD flash storage, you need an USB to SD flash storage adapter. Most of the USB to SD flash storage adapter in the market work well.

Work through the following steps to configure the SD flash storage with BIOSLoader and EBOOT.BIN:

1. Create the following directory on your development station:

   C:\MyTemp

2. Copy the DiskPrep.exe executable to the C:\MyTemp folder.

3. Copy the EBOOT.BIN Ethernet bootloader file to the C:\MyTemp folder.

4. With the SD flash storage inserted to the USB to SD adapter, attached the adapter to the development station’s USB interface.

5. After the development station detected and recognized the SD flash storage, launch the DiskPrep.exe executable, using “Run as administrator”.

   - From the file explorer, navigate to the C:\MyTemp folder.
   - Right mouse click on the DiskPrep.exe executable and select “Run as administrator”.
   - When the User Account Control message box popup, click on Yes to launch DiskPrep, as shown in Figure A1.
6. From the Disk Selection, select the detected SD flash storage.

7. From the Use File System Format section, select FAT or FAT32. Note: FAT file system is limited to 2 GB file partition.

8. For the Splash Screen selection, select the “Use the default Windows CE splash” option.

9. For the BOOT.INI selection, select the “Place a BOOT.INI file on the disk” option.

10. Select the last option, select the “Load specific image file copied from” option and click on Browse to bring up a file selection dialog.

    - From the file selection dialog, navigate to the C:\MyTemp folder, select EBOOT.BIN and click Open.

10. From the DiskPrep program screen, click OK to format the SD storage.

11. At this point, the Confirm Dangerous Action screen popup with a warning message. Click on OK to continue. As it continuing, a DiskPrep progress screen popup to display progress.

12. When finish, the Success screen popup and display a message, “Disk Preparation Completed ok”, to indicate the task is completed successfully.

The SD flash Storage is now configured with BIOSLoader to launch EBOOT.BIN, with a default startup splash screen. The following files are copied to the SD flash storage:

- BLDR (BIOSLoader)
- BOOT.INI (Boot configuration file)
- EBOOT.BIN (Ethernet bootloader)
- SPASH.BMX (Graphic file for the startup splash screen)
At this point, the SD flash storage is configured to boot with BIOSLoader. During startup, the BIOSLoader is configured, via the BOOT.INI configuration file, to launch EBOOT.BIN (an Ethernet bootloader).

When launched on the target device, the EBOOT.BIN bootloader broadcast a series of BOOTME messages, attempting to establish connectivity to a Platform Builder development station and download OS runtime image.

**Modify BOOT.INI Configuration File**

The BOOT.INI configuration file can be modified to perform the following:

- Launch a Compact 7 OS runtime image, NK.BIN, after a configured time delay.
- Before the configured delay time expired, press “1” to launch the Ethernet bootloader, EBOOT.BIN, to download OS runtime image from the development station.
- Before the configured delay time expired, press “2” to launch a secondary Compact 7 OS runtime image, NK2.BIN.

You can use a text editor, such as NotePad, to view the BOOT.INI configuration file on the SD flash storage. Here are the entries in the BOOT.INI file created from the earlier exercise in this appendix:

```plaintext
# BIOS loader configuration file
# Primary File to Load on startup:
BinFile=EBOOT.BIN

# Device name root string:
DeviceNameRoot=CEPC

# Video Setting:
Video=on

# Physical screen width
# Must be >= DisplayWidth
PhysicalWidth=640

# Physical screen height
# Must be >= DisplayHeight
PhysicalHeight=480

# Display Depth
# Possible values are 8, 15, 16, 24, 32
DisplayDepth=16

# Display width
DisplayWidth=640

# Display height
DisplayHeight=480

Note: Lines begin with the “#” character are comments.
```
Change the BOOT.INI file with the following entries:

```
# Boot delay (in seconds) in which to select alternative image
Delay=10

# Primary File to Load on startup:
BinFile=NK.BIN

# Alternative image
# While booting hit the key corresponding to the suffix char (e.g. \"1\" for BinFile1) to boot the alternative image
BinFile1=EBOOT.BIN

# Device name root string:
DeviceNameRoot=CEPC

# Video Setting:
Video=on

# Physical screen width
# Must be >= DisplayWidth
PhysicalWidth=640

# Physical screen height
# Must be >= DisplayHeight
PhysicalHeight=480

# Display Depth
# Possible values are 8, 15, 16, 24, 32
DisplayDepth=16

# Display width
DisplayWidth=640

# Display height
DisplayHeight=480
```

Save the above BOOT.INI file to the SD flash storage.

In addition, copy the NK.BIN runtime image file to the SD flash storage, from the OS design, in the following directory:

```
C:\WINCE700\OSDesign\<OS design name>\<OS design name>\RelDir\<BSP name>_Release
```

For the 3350MX_Projector project, copy the NK.BIN runtime image from the following directory:

```
C:\WINCE700\OSDesign\3350projector\3350Projector\RelDir\ICOP_eBox3350MX_70C_x86_Release
```
Appendix-B: Windows Network Projector User Scenarios

The Windows Embedded Compact technologies behind WNP's enable you to build a number of different device types.

Windows Network Projector Usage Scenario

The following diagram shows a direct implementation of WNP functionality built into a new or existing projector design. With this integrated support, the projector provides the capability of being discovered and connected to by a Windows Vista or Windows 7-based PC. The example in the following diagram shows the WNP device used with an infrastructure network connection.

The example in the following diagram shows the WNP device used with an ad hoc network connection.
Windows Network Projector Adapter

The following diagram shows a WNP adapter. This WNP device can be used to retrofit an existing conference room projector and extend its current functionality to that of a WNP. The WNP adapter device provides the seamless discovery and connection to a Windows Vista or Windows 7 based PC and remotely displays the PC desktop.
Appendix-C: Hardware Platform Considerations

The selected hardware platform for the Windows Network Projector device must meet certain performance to deliver positive user experience.

**Hardware: Graphics Considerations**

WNP system performance is highly dependent on the graphics and networking hardware’s performance. Graphics hardware and networking hardware are the most critical hardware components for achieving optimal performance in a WNP device’s display capabilities. Display performance for a WNP device is directly dependent upon the speed at which the target device hardware is able to process and render the image and graphics changes.

The graphics hardware’s system-to-video data transfer rate has the greatest impact on display and rendering performance. Graphics hardware with a system-to-video data transfer rate of 60 MB/s or greater should result in reasonable performance for most presentations with graphical animation. Using graphics hardware with higher system to video data transfer rates will improve performance.

At a minimum, a WNP device must support a display resolution of 800 x 600, also known as SVGA resolution, or better. The network projector’s maximum resolution is limited by the supported Compact 7 and graphics hardware resolutions. Support for higher display resolutions and increased color depth will increase the amount of data that needs to be processed, which will affect the time and system resources necessary to render display updates.

**Hardware: Microprocessor Considerations**

A Compact 7 based WNP device requires a modern 32-bit microprocessor. Compact 7 supports various microprocessors from the ARM, MIPS and x86 families.

The microprocessor must support 32-bit addressing, and because Compact 7 implements a virtual memory system, the microprocessor must contain a memory management unit (MMU) to support virtual-memory-to-physical-memory translations.

In addition to the main microprocessor, consider graphic processor or digital signal processor (DSP) as part of the design to offload graphic processing from the microprocessor which enables the hardware platform to use a slower microprocessor.

The following list shows important application-specific considerations for choosing a microprocessor:

- Targeted Performance
- Processing power
- Instruction set
- Clock speed
- Power consumption
- Heat dissipation

**Hardware: System Memory Considerations**

ROM and RAM requirements for Compact 7-based WNP devices depend on several significant factors, such as CPU selection, applications implemented, and performance considerations.
**RAM**
The WNP software and the shell together require at least 64 MB of RAM. This requirement does not reflect the amount of RAM that is required for the graphics buffer or the OEM-supplied software. This memory requirement supports a WNP running one or two RDP sessions. The recommended amount of RAM for a WNP is 128 MB.

For execution of software on a Compact 7-based WNP device, your hardware must have 32-bit wide SDRAM with a minimum frequency of 100 MHz. To provide a better user experience with smoother graphics output, use more SDRAM to buffer more graphics data. Additionally, using more SDRAM allows more applications to reside on the device.

**ROM or Flash**
WNP requires sufficient nonvolatile memory, such as flash memory or nonvolatile RAM, to store the operating system image, application software, universally unique identifier (UUID) and the registry entries. A WNP device requires at least 32MB of ROM, plus user storage.

Additionally, the ROM must be sufficient to support software that you add to the network projector or display image. Microsoft recommends that you provide sufficient nonvolatile memory to store a minimal image backup as well as provide room for the registry to grow to at least 50 KB in size.

**Hardware: Storage Considerations**
Depending on the selected hardware platform and design objective, additional storage can be added to enhance the device's function and feature.

You can optionally support internal flash memory storage. Internal flash memory storage can store the runtime image, data files for the device, or both.

If the internal flash memory is the storage medium for the runtime image, the boot loader must be able to download another copy of the runtime image by using another storage medium such as a network or a USB key drive, or a hard disk. This ensures recovery of the boot loader and the runtime image from possible storage corruption. Use this same storage medium to update the runtime image.

Examples of internal flash memory storage are the following:

- Disk on module (DOM)
- Disk on chip (DOC)
- NAND flash array
- NOR flash array

**Hardware: Networking Interface Considerations**
For wireless connectivity, WNP devices must support the specified protocols to interact with an 802.11-compliant Windows 7-based PC. Specifically, support for WPA2-compliant 802.11g networking is required. A premium wireless projector device should additionally support the IEEE 802.11a wireless standard. Support for wired networking is optional. Two networking cards are required within, or connected to, a network projector to enable it to simultaneously connect to both a wireless Windows 7-based PC and a LAN.
Wireless Networking

802.11a
IEEE 802.11a operates at a data transmission rate as high as 54 megabits per second (Mbps) and uses a radio frequency of 5.8 GHz. Instead of DSSS, 802.11a uses orthogonal frequency-division multiplexing (OFDM). OFDM allows data to be transmitted by sub-frequencies in parallel. This modulation mode provides better resistance to interference and improved data transmission. This higher-speed technology improves WLAN networking performance for video and conferencing applications and for future streaming video support.

802.11b
IEEE 802.11b, an enhancement to IEEE 802.11, provides standardization of the physical layer to support higher bit rates. IEEE 802.11b uses 2.45 GHz, the same frequency as IEEE 802.11, and supports two additional speeds, 5.5 Mbps and 11 Mbps. It uses the DSSS modulation scheme to provide higher data transmission rates. The bit rate of 11 Mbps is achievable in ideal conditions. In less-than-ideal conditions, the slower speeds of 5.5 Mbps, 2 Mbps, and 1 Mbps are used.

802.11g
IEEE 802.11g is an extension to 802.11b and supports bandwidth up to 54 Mbps, using the 2.4 GHz frequency for greater range. 802.11g is backwards compatible with 802.11b, meaning that 802.11g access points will work with 802.11b wireless network adapters and vice versa. When there is contention for bandwidth, 802.11g falls back to 802.11b to allow for 802.11b devices to operate. The presence of an 802.11b participant significantly reduces the speed of an 802.11g network impacting the performance of wireless data transfer.

Wired Networking
Support for wired networking is optional for WNPs. Providing support for a wired network connection for a Compact 7 based WNP device offers the option to connect to a corporate LAN. A wired interface will require a 10/100 Mbps 10BaseT/100BaseT (IEEE 802.3/802.3u) network connection. To reduce costs, you can attach an Ethernet connector or Ethernet header.

If needed, you can connect an Ethernet debug port to the Ethernet connector or Ethernet header for development and debugging purposes. Attaching an Ethernet connector or Ethernet header allows you to save the cost of including an Ethernet debug port in every hardware platform. The following list shows the requirements for the network controller:

- Auto-detection and auto-switching between the 10 Mbps and the 100 Mbps data transfer rates.
- Detection when the network cable is disconnected.
- Support for direct memory access (DMA) transfers to system memory up to the maximum bandwidth allowable by the IEEE 802.3/802.3u specification.

Hardware: Debugging Considerations
To debug a WNP, you must have an Ethernet debug port on the development board that supports 10/100 megabits per second (Mbps) 10BaseT/100BaseT (802.3). This Ethernet connection is independent of the end-user 10/100 Mbps 10BaseT/100BaseT Ethernet connection that can be added to the network projector functionality.
For additional debugging capabilities, you can add a serial debug port or a USB debug port. These debug ports are optional and are independent of any other end-user serial ports or USB ports.

**Hardware: Real-Time Clock Considerations**
The real-time clock hardware manages the date and the time. For informational purposes, the user interface (UI) displays the date and the time.

The real-time clock must have an alarm and a backup battery. When appropriate, the alarm generates events to wake the system from standby. The dynamic password functionality uses the real-time clock to enable updates.

**Hardware: Data Bus Considerations**
Network projectors can support various types of buses. Use either the device drivers that are provided with the network projector, or use the device drivers that are provided by a third party.

- Inter-Integrated Circuit (I2C) or Inter-IC Sound (I2S)
- PCI
- Secure Digital Input/output (SDIO)

**Hardware: Board Support Package Considerations**
To build a WNP, your BSP must support Windows Embedded Compact 7. If you already have an existing network projector platform built on a previous version of Windows Embedded CE then you must upgrade that platform to the latest version. For a list of the BSPs supported in the current version of Windows Embedded Compact, visit the Supported Board Support Packages website on MSDN:


When considering the microprocessor and BSP for your WNP, discuss the availability with Microsoft and the microprocessor hardware manufacturer.

Make sure to consider all the requirements that are specific to your hardware platform and BSP when developing Compact 7-based WNP devices. Base your development decisions on your specific target hardware.

**Hardware: Additional Components**
You can increase the functionality of your WNP design by adding support for additional hardware features. The following list shows some possible options.

- Keyboard
- Remote Control
Appendix-D: Windows Network Projector Display Resolution

The display resolutions available to a WNP are based on the capabilities of the display hardware and driver. You can place limits on the display resolutions that a WNP can actually use by changing settings in the registry.

The minimum possible display resolution for a WNP is 800 x 600 with a color depth of 8 bpp. The maximum resolution is limited by the network projector's display hardware. The higher the resolution and color depth, the greater the performance expense for the projector.

During the connection phase, the WNP service negotiates with the Windows Vista-based PC to connect with the closest supported resolution through the following steps:

1. The network projector service queries for the supported resolutions from the display driver.
2. The supported resolutions are then compared against the maximum resolution, as defined in the network projector’s registry settings.
3. The supported resolutions that are less than or equal to the maximum resolution are sent to the Windows Vista Projector Connection Wizard.
4. The Windows Vista-based PC compares the current resolution of the PC to the resolutions supported by the WNP.
5. If the Windows Vista-based PC's resolution is equal to or less than the supported WNP resolutions, then the Windows Vista-based PC connects to the Network Projector.
6. If not, then the user will be notified with an error message that the user must reduce to the Windows Vista-based PC's resolution and attempt to connect again.

You can set the maximum display resolution of the WNP through the `HKEY_LOCAL_MACHINE\Software\Microsoft\PictorService` registry key.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxScreenWidth</td>
<td>The default value is 1024. The maximum horizontal resolution of the WNP's display, in pixels.</td>
</tr>
<tr>
<td>MaxScreenHeight</td>
<td>The default value is 768. The maximum vertical resolution of the WNP's display, in pixels.</td>
</tr>
</tbody>
</table>
Appendix-E: Connecting to a Windows Network Projector

The diagram below shows an example of connecting WNP device, the Platform Builder development computer and a Windows 7-based PC for testing.

To Connect to a Windows Network Projector:

1. Connect the PC running Platform Builder, the Compact 7 target device, and the Windows 7-based PC on an IP network.

2. Download the WNP OS runtime image to the Compact 7 target device.
   The Network Projector splash screen should display the following information:
   - **Projector**: <projector name>
   - **Password**: <the dynamic password displayed>
   - **URL**: <the IP address assigned to the Compact 7-based device>

3. On the Windows 7-based PC, ping the address of the Compact 7 target device through the command window.

4. From the **Start** menu on the Windows 7-based PC, choose All Programs, choose Accessories, and then choose **Connect to a Network Projector**.

5. Choose **Search for a projector**.

6. Once the Windows 7-based PC has detected the projector, select device named CEProjector and enter the password as it is displayed on the splash screen.

7. Choose **Next**.

8. Leave the connection type as Mirrored and choose **Connect**.
To Troubleshoot the Connection

1. The splash screen does not display an IP address.
   
   Check to ensure the Compact 7 OS runtime image is configured to enable DHCP.
   
   Did you connect the Compact 7 target device to a network switch or router that supports DHCP?

2. The Windows 7-based PC cannot find the Compact 7 WNP device.
   
   Does the WNP device have an IP address?
   
   Does the Windows 7-based PC have an IP address?

3. The Windows 7-based PC can find the projector but cannot connect to it.
   
   Did you try rebooting the Compact 7-based device?
   
   Did you enable the exemption in the Firewall for the program Connecting to a Network Projector?
   
   Did you try stopping IPSec? To do this, open a command window in administrator mode and enter **net stop bfe**.
Appendix-F: Windows Network Projector Registry Settings

The registry settings files for WNP s are located in:

%WINCEROOT%\PUBLIC\RDP\OAK\FILES

Sample Network Projector Application Settings
The registry key HKEY_LOCAL_MACHINE\init sets the startup priority for the WNP application.

<table>
<thead>
<tr>
<th>Value : type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch80 : DWORD</td>
<td>The default setting is &quot;pictorapp.exe&quot;. Identifies the name of the WNP application.</td>
</tr>
<tr>
<td>Depend80 : hex</td>
<td>The default value is hex:14,00,1e,00, 3c,00. Defines the startup dependencies for the WNP application</td>
</tr>
</tbody>
</table>

For more information about launching an application when the OS starts up, see Configuring a Registry File to Run an Application at Startup.

Network Projector Service Settings
You can configure the WNP service settings with the following registry keys:

HKEY_LOCAL_MACHINE\Software\Microsoft\PictorService

<table>
<thead>
<tr>
<th>Value : type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProjectorName : DWORD</td>
<td>The default value is &quot;CeProjector&quot;</td>
</tr>
<tr>
<td>AdhocSSID : DWORD</td>
<td>The default value is &quot;CeProjector&quot;</td>
</tr>
</tbody>
</table>