Vortex86DX-SPARK
Windows Embedded CE 6.0 R3
Jump Start Guide

Rev 2.5

By Samuel Phung, Windows Embedded MVP
ICOP Technology Inc.

This guide includes hands on exercises to develop custom Windows Embedded CE 6.0 OS runtime image, native and managed code application, with step-by-step instruction.

[Blank page]

Introduction

This step-by-step guide, using hands-on approach, is created to show how to develop, customize, build a Windows Embedded CE 6.0 R3 (CE 6.0) OS design, generate a CE 6.0 OS runtime image from the OS design, download the runtime image to a target device and steps to develop managed and native code application for the CE 6.0 devices. The following subjects are covered in this jump start guide:

- What's included in the jump start kit.
- Windows Embedded CE 6.0 and development environment overview.
- Required software and recommended installation sequence.
- Develop and customize a CE 6.0 OS design project.
- Generate a custom CE 6.0 OS runtime image from the OS design project, establish connectivity and download the runtime image to a target device.
- Generate a CE 6.0 OS runtime image with KITL (Kernel Independent Transport Layer), and use remote tools to debug the runtime image on a target device.
- Develop CE 6.0 Managed code application in C# using Visual Studio 2005/2008. Establish connectivity between the target device and Visual Studio 2005/2008 development workstation using CoreCon and download the application to the device.
- Develop CE 6.0 native node application in C++ using Visual Studio 2005/2008. Establish connectivity between the target device and Visual Studio 2005/2008 development workstation using CoreCon and download the application to the device.
- CE 6.0 information and technical resources.

This jump start guide does not cover programming concepts and development techniques. The primary objective for this guide is to provide information to help developer new to the Windows Embedded CE development environment to become familiar with Windows Embedded CE 6.0 development, tools and steps to perform common development tasks.

The exercises in this guide are tested on Windows XP Professional and Windows Vista development workstations using Visual Studio 2005 Professional, CE 6.0 Platform Builder, with R2 and R3 releases installed, to develop the OS design project and generate the CE 6.0 OS runtime image. Both Visual Studio 2005 and 2008 are used to develop the sample managed and native code applications.

- On the Windows XP Professional SP3 development workstation, Visual Studio 2008, Visual Studio 2005, Visual Studio 2005 SP1, CE 6.0, CE 6.0 SP1, CE 6.0 R2, CE 6.0 R3 ICOP_VDX6326_60B Board-Support-Package and VDX6326_WINCE600_SDK are installed.
- On the Windows Vista SP1 development workstation, Visual Studio 2008, Visual Studio 2005, Visual Studio 2005 SP1, Visual Studio 2005 SP1 Update for Vista, CE 6.0, CE 6.0 SP1, CE 6.0 R2, CE 6.0 R3, ICOP_VDX6326_60B Board-Support-Package and VDX6326_WINCE600_SDK are installed.

The VDX-6326 Single-Board-Computer (SBC) is used as the target device for the exercises in this guide. Detailed information about the SBC is available in appendix D. The exercises in this guide are created with both the development workstation and the SBC connected to the same Local Area Network with DHCP service to provide IP addresses dynamically.

When working in a development environment without DHCP service, configure the development workstation and the SBC with static IP addresses using the same subnet; refer to appendix B for more information.

Additional references and technical information resources for Windows Embedded CE are listed in Appendix C.

To go through the exercises in this guide, Visual Studio 2005, Visual Studio 2005 Service Pack 1, Visual Studio 2005 Service Pack 1 update for Vista, Windows Embedded CE 6.0 Platform Builder, Windows Embedded CE 6.0 Service Pack 1, Windows Embedded CE 6.0 R2, R3, ICOP_VDX6326_60B Board-Support-Package and VDX6326_WINCE600_SDK are needed. Refer to Appendix A & B for device preparation and setup information.

******** Notes ********

During the Visual Studio 2005 and CE 6.0 Platform Builder installation process, the installation wizard provides the option to install the software to a different location from the default. For the most part, with Visual Studio 2005 and CE 6.0 Platform Builder installed to non-default directories, both the Visual Studio and Platform Builder IDE are able to function just fine.

However, some of the CE 6.0 3rd party components are created with reference to the default CE 6.0 installation folder, with hard coded path. These components will not work as expected. Although it's possible to modify the component's configuration file to enable the component to function properly, it's not within the scope for this jump start guide to cover this subject.

To insure you can work through all of the exercises in this guide. Please install all software to the default installation folder. The sample projects, Board-Support-Package, utilities and sample codes provided as part of this jump start kit are created on a workstation with Visual Studio and Platform Builder installed to the default installation folder. Some of the components and build processes may not work properly on workstation with the development software installed to a folder different from the default installation folder.

Vortex86DX-SPARK Jump Start Kit

The Vortex86DX-SPARK Jump Start kit includes the following

- VDX-6326, an 800 MHz Single-Board-Computer (SBC) built with the following features:
  - 800 Mhz Vortex86DX System-On-Chip
  - XGI Z9s Video with VGA output and support for TFT/LVDS LCD
  - 256MB soldered on DDR2 system memory
  - Enhanced IDE (UltraDMA-100/66/33)
  - CM119 USB Audio
  - Onboard 4MB SPI flash
  - Three 10/100Mbps Ethernet
  - Three USB 2.0 Host interfaces
  - Three RS-232 Serial ports, one RS-232/422/485 Serial port
  - Parallel port
  - 16-bit GPIO (Can be configured as PWM)
  - Mini-PCI expansion slot
  - Type I/II CF Card (Support storage card only)
  - PC/104 expansion slot
  - PS/2 keyboard & mouse
  - Two watchdog timers (Programmable from 30.5µs to 512 seconds)
- Enclosure to house the VDX-6326 SBC and extend the I/O peripherals from the SBC using common connectors.
- 512 MB EmbedDisk (IDE bootable flash storage)

  The 512 MB EmbedDisk is pre-configured to boot to DOS and provides a DOS menu with options to load a prebuilt CE 6.0 OS runtime image from the local storage or launches the included Ethernet boot loader to download a CE 6.0 OS runtime image from a connected CE 6.0 Platform Builder development workstation.
- Windows Embedded CE 6.0 SPARK software kit, which includes the following:
  - Windows Embedded CE 6.0 R2

    Windows Embedded CE 6.0 R3 update is available for download from the following URL: http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=bc247d88-ddb6-4d4a-a595-8eee3556fe46

    Note: CE 6.0 R3 is not required to work through the exercises in this guide. However, installing CE 6.0 R3 will provide additional useful components, which you can include to the OS design project and enhance the function and features.
  - Visual Studio 2005 Professional
- Windows Embedded CE 6.0 jump start CD – This jump start CD includes the following:
  - Board-Support-Package for Windows Embedded CE 6.0
  - SDK for Windows Embedded CE 6.0
  - Ethernet boot loader
  - CoreCon component for Windows Embedded CE 6.0
  - AutoLaunch component for Windows Embedded CE 6.0
  - RegFlushApp component for Windows Embedded CE 6.0
  - Sample project codes for the exercises in this guide

Note: The Board-Support-Package and CE 6.0 components provided on the jump start CD support CE 6.0, CE 6.0 R2 and CE 6.0 R3 development.

- One RJ-45 Ethernet Crossover cable

  With a crossover Ethernet cable and proper static IP address settings, a Windows Embedded CE device can be connected directly to the development workstation to create a standalone development environment.

- One RS-232 null serial cable for serial debug

  The RS-232 null serial cable is used to establish connectivity between one of the serial port on the Windows Embedded CE device to an available serial port on the development workstation. With proper configuration, debug messages from the Windows Embedded CE device can be captured by the development workstation using Hyper Terminal.

  The default serial port setting used for this option is:

  38400 Baud, 8 data bits, no parity and 1 stop bit.

******* Reading Notes *******

For the contents in this jump start guide, “Windows Embedded CE 6.0”, “Windows Embedded CE 6.0 R2” and “Windows Embedded CE 6.0 R3” are abbreviated as “**CE 6.0**”.

Board-Support-Package is abbreviated as “BSP”.

“VDX-6326 Single-Board-Computer” is abbreviated as “SBC”.

Visual Studio 2005 is abbreviated as VS2005. Visual Studio 2005 Integrated-Development-Environment is abbreviated as VS2005 IDE.

Visual Studio 2008 is abbreviated as VS2008. Visual Studio 2008 Integrated-Development-Environment is abbreviated as VS2008 IDE.

“Target device” is a common term used in many of the CE 6.0 help document and application notes. The “target device” is a general term referring to the hardware device used for CE 6.0 development. The VDX-6326 SBC is the target device for the exercises in the guide.

Table of Contents

Part 1 – What's New

Windows Embedded CE 6.0 R3

Windows Embedded CE 6.0 R3 (CE 6.0) is a hard Real-time operating system with ability to handle 32,000 concurrent processes and 2GB memory footprint for each process.

CE 6.0 delivers reliable, secure performance in a small footprint along with the latest networking, multimedia and communications technologies. CE 6.0 provides developers with broad range of device support with enhanced features, including robust file-system, Web services for device, voice over IP (VoIP) phone, network gateway configurations, platform development tool enhancements, greater application compatibility with other Windows Embedded CE based devices, Internet Explorer, Windows Media CODECs, Microsoft .NET Compact Framework, and a number of other newly supported protocols and services.

In addition to improvement to the OS, the CE 6.0 R3 release added the following components:

- Silverlight for Windows Embedded
- Flash Lite Browser plug-in to render Flash contents
- Touch and Gesture
- Microsoft Office and PDF viewers
- QQ Messenger
- Connection Manager

Combining large pool of production quality device drivers, programming libraries and efficient development tools, CE 6.0 provides the ideal Rapid Application Development environment to create the next generation of smart, media rich, connected and service oriented devices.

For more information about CE 6.0, visit http://www.microsoft.com/windowsembedded/en-us/products/windowsce/default.mspx.

Windows Embedded CE 6.0 R3 Development Tools

Platform Builder is the development tool used to develop/configure the OS design and generate runtime image from the OS design. The latest version of Windows Embedded CE 6.0 R3 Platform Builder is a Plug-in for the Visual Studio 2005 Integrated-Development-Environment (IDE) and takes full advantage of the efficient development environment provided by the Visual Studio 2005 IDE.

Visual Studio 2005 SP1 is needed to develop application for Windows Embedded CE 6.0.

For Windows Vista, in addition to Visual Studio 2005 SP1, Visual Studio 2005 SP1 Update for Vista is also needed.

To minimize potential problem, it's best to install all of the available service pack, QFEs and updates in chronological order.

To locate available service pack, QFE and update, search Microsoft download using the "Windows Embedded CE 6.0" key words from the following URL:

http://www.microsoft.com/download/

Part 2 – Development Environment Overview

Windows Embedded CE 6.0 – Platform Builder

Platform Builder is the development tool used to develop and configure CE 6.0 OS design project. Customized CE 6.0 OS runtime image, developed to support specific target device, is generated from OS design project. It also provides the remote tools to debug CE 6.0 OS runtime image on the target device.

For the CE 6.0 release, Platform Builder is a plug-in to the Visual Studio 2005 Integrated Development Environment (IDE), and requires Visual Studio 2005 to function.

Here are the typical steps to develop a CE 6.0 solution after the hardware platform is selected:

- Develop CE 6.0 BSP for the selected hardware.

  Note: Some company charges a fee for the BSP and requires licensing fee for each device using the OS runtime image generated by using the BSP.

  ICOP Technology provides the BSP at no cost. ICOP customer can use the BSP freely to generate OS runtime image, and distribute OS runtime image generated by using ICOP BSP on ICOP hardware for the distribution without licensing fee.

  Customer is obligated to secure appropriate license agreement from Microsoft and pay CE 6.0 runtime license fees for the OS runtime image distributed as part of the solution.

- Develop an OS design project using Platform Builder within the Visual Studio 2005 IDE.
- Customize and configure the OS design project with the desired features and functions.
- Generate the CE 6.0 OS runtime image using Platform Builder within the Visual Studio 2005 IDE.
- Establish connectivity between the target device and the Visual Studio 2005 IDE to download the CE 6.0 OS runtime image to the target device for testing and debug.
- Generate a CE 6.0 SDK from the OS design to support application development.
- Develop managed or native code CE 6.0 application for the target device.

  Note: Application development can take place concurrently as the hardware and OS runtime image being fine tuned.

- After the satisfied CE 6.0 OS runtime image and application are created, deploy the solution to the target device's local storage with an appropriate boot loader to launch the OS when the device power on.

Note: The above scenario assumes a functional Board-Support-Package is available for the target device.

Windows Embedded CE 6.0 – Remote Tools

Using remote tools provided as part of the Platform Builder development environment, developer is able to debug a CE 6.0 OS runtime image, built with KITL (Kernel Independent Transport Layer) enabled and downloaded from the Visual Studio 2005 IDE to the target device.

The following remote tools are included with the CE 6.0 Platform Builder:

- Remote Call Profiler
- Remote File Viewer
- Remote Heap Walker
- Remote Kernel Tracker
- Remote Performance Monitor
- Remote Process Viewer
- Remote Registry Editor
- Remote Spy
- Remote System Information
- Remote Zoom-in

Microsoft Visual Studio 2005 and 2008

There are multiple options to develop Windows Embedded CE applications.

- Develop Native code application with C programming language, using the Platform Builder IDE.
- Develop Native code application with Visual C++, using the Visual Studio 2005 IDE.
- Develop Managed code application with Visual C#, using the Visual Studio 2005 IDE.
- Develop Managed code application with Visual Basic, using the Visual Studio 2005 IDE.
- Develop Native code application with Visual C++, using the Visual Studio 2008 IDE.
- Develop Managed code application with Visual C#, using the Visual Studio 2008 IDE.
- Develop Managed code application with Visual Basic, using the Visual Studio 2008 IDE.

A Software Development Kit (SDK), generate from the OS design for the target device, is needed to support application development using Visual Studio 2005 and 2008.

The Visual Studio (2005 and 2008) IDE provides an efficient development environment, making it possible to download the application from the Visual Studio IDE to the target device for testing and debug using CoreCon connectivity.

Using the Visual Studio IDE with CoreCon connectivity, application developer is able download application to a CE 6.0 target device, launches the application, set breakpoint, steps through and executes the code one-line-at-a-time on the target device.

Important Note – Do not use "Build and Sysgen" or "Rebuild and Clean Sysgen"

To generate CE 6.0 OS runtime image from the OS design, don't use the **Build and Sysgen** and **Rebuild and Clean Sysgen** options. When one of these options is executed, it deletes and recompiles the binary files, required by the build process to generate the OS runtime image.

Not all source codes are provided to recompile the required binary files. Without the source codes needed to generate the required binary files, the system will failed to complete the build process to generate the OS runtime image.

To avoid accidently click on these build options, it's recommended to remove them from the Visual Studio menu. To remove these two build options, with an OS design project active, work through the following steps:

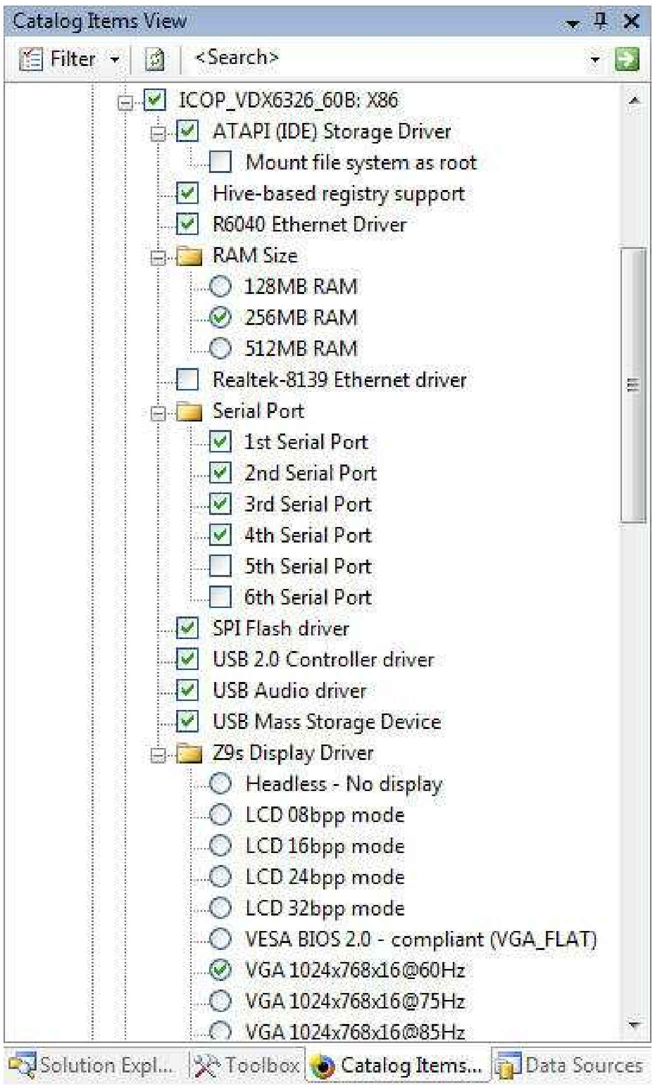
- From VS2005 IDE, select **Tools | Customize…** to bring up the Customize dialog.
- With the Customize dialog open, from the VS2005 IDE, click on **Build | Advanced Build Commands | Build and Sysgen** and drag it out to an empty space on the VS2005 IDE to remove it from the menu.
- Repeat the above step to remove **Build | Advanced Build Commands | Rebuild and Clean Sysgen** from the menu.

The Target Device – VDX-6326 Single-Board-Computer

The VDX-6326 SBC that comes with this jump start kit is built with the following features:

- Video with support for VGA, TFT-LCD and LVDS-LCD
- Audio with microphone input and line-out
- Enhanced IDE port (UltraDMA-100/66/33)
- Compact Flash (storage device only)
- FDD port
- Three USB 2.0 host interface
- Three 10/100 Mbps Ethernet
- Three RS-232 serial ports
- RS-232/422/485 serial port
- Parallel port
- PC/104 expansion bus
- Mini-PCI expansion slot
- 16-bit GPIOs (Can be configured as digital-input, digital-output or PWM)
- Two Watchdog timers (programmable from 30.5 µs to 512 seconds)

The CE 6.0 BSP provided along with this kit includes the necessary hardware adaptation code, device driver and configuration files needed to generate the CE 6.0 OS runtime image, as shown in Figure 01.



- [✓] ICOP_VDX6326_60B: X86
  - [✓] ATAPI (IDE) Storage Driver
    - [ ] Mount file system as root
  - [✓] Hive-based registry support
  - [✓] R6040 Ethernet Driver
  - RAM Size
    - ○ 128MB RAM
    - ✓ 256MB RAM
    - ○ 512MB RAM
  - [ ] Realtek-8139 Ethernet driver
  - Serial Port
    - [✓] 1st Serial Port
    - [✓] 2nd Serial Port
    - [✓] 3rd Serial Port
    - [✓] 4th Serial Port
    - [ ] 5th Serial Port
    - [ ] 6th Serial Port
  - [✓] SPI Flash driver
  - [✓] USB 2.0 Controller driver
  - [✓] USB Audio driver
  - [✓] USB Mass Storage Device
  - Z9s Display Driver
    - ○ Headless - No display
    - ○ LCD 08bpp mode
    - ○ LCD 16bpp mode
    - ○ LCD 24bpp mode
    - ○ LCD 32bpp mode
    - ○ VESA BIOS 2.0 - compliant (VGA_FLAT)
    - ✓ VGA 1024x768x16@60Hz
    - ○ VGA 1024x768x16@75Hz
    - ○ VGA 1024x768x16@85Hz

Fig. 01 – ICOP_VDX6326_60B BSP for CE 6.0

To provide support for the VDX-6326 SBC's GPIO and Watchdog timers, the following DLLs are provided:

- GPIO.dll (To access the 16-bit GPIOs)
- WDT.dll (To access the Watchdog timers)

When developing an OS design with the ICOP_VDX6326_60B BSP, these DLLs are included in the OS runtime image by default.

Sample native and managed code projects showing how to use these DLLs are provide on the jump start CD.

For more information about the VDX-6326 SBC's GPIO, Watchdog timers and the supporting DLLs, refer to Appendixes O and P.

Part 3 – Jump Start Kit Software & Installation

The following software packages are provided with the Vortex86DX-SPARK jump start kit:

- Windows Embedded CE 6.0 SPARK software kit which includes the following:
  - Visual Studio 2005 Professional
  - Windows Embedded CE 6.0 R2

    Windows Embedded CE 6.0 R3 update is available for download from the following URL:

    http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=bc247d88-ddb6-4d4a-a595-8eee3556fe46

    Note: CE 6.0 R2 and R3 are not required to work through the exercises in this guide. However, installing CE 6.0 R2 and R3 will provide additional useful components, which you can include to the OS design project to enhance function and feature.
- ICOP_VDX6326_60B Board-Support-Package for the VDX-6326 SBC.
- VDX6326_WINCE600_SDK Software Development Kit for the VDX-6326 SBC.
- CoreCon component (CoreCon_v200_x86) needed to establish connectivity between the CE 6.0 target device and Visual Studio IDE
- AutoLaunch component (AutoLaunch_v200_x86), an utility for launching one or more application automatically when CE 6.0 start
- RegFlushApp component, a CE 6.0 utility to flush and save changes to the system registry when Hive-based registry is implemented.
- Sample project codes for the exercises in this guide
- Sample project codes showing how to access and use the Vortex86DX GPIO and Watchdog timer

Visual Studio 2005 Professional

Visual Studio 2005 (VS2005) is a popular developer friendly development tool for developing broad range of applications that run on different version of the Windows Operating Systems.

Using the VS2005 development tools, developer can create applications for the following Windows Operating Systems:

- Windows 7
- Windows Vista
- Windows XP
- Windows 2008 Server
- Windows 2003 Server
- PocketPC
- Windows Mobile Smartphones
- Windows Embedded CE devices

Windows Embedded CE 6.0 Platform Builder

Windows Embedded CE 6.0 Platform Builder is a plug-in to the VS2005 IDE, and takes advantage of the efficient VS2005 IDE tools. The latest release, Windows Embedded CE 6.0 R3, is an incremental release to provide additional features and technologies. For the purpose of this guide, we will refer to “Windows Embedded CE 6.0”, “Windows Embedded CE 6.0 R2” and “Windows Embedded CE 6.0 R3” as “CE 6.0”.

Note: The software provided as part of the Windows Embedded CE 6.0 SPARK software kit is licensed for personal, academic and non-commercial use only. When using CE 6.0 software for commercial application, please purchase the software from a qualified distributor.

The CE 6.0 R3 release is available for download from the following Microsoft website:

http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=bc247d88-ddb6-4d4a-a595-8eee3556fe46

Board-Support-Package

Board-Support-Package (BSP) consists of all necessary CE 6.0 device drivers, hardware adaptation library and configuration files to support the targeted device, and is needed by CE 6.0 Platform Builder to develop the OS design and generate CE 6.0 OS runtime image for the targeted device. The `ICOP_VDX6326_60B.msi` BSP is provided as part of this jump start kit to develop OS design and generate CE 6.0 OS runtime image for the VDX-6326 SBC. This BSP is provided with the jump start kit CD in the `\Software` folder.

CoreCon Connectivity

CoreCon is needed to establish connectivity between the CE 6.0 target device and the Visual Studio (2005 and 2008) IDE for the purpose of developing application and downloading the application onto the target device for testing and debug. The CoreCon files are installed to the development workstation as part of the VS2005 and VS2008 installation to the following folder on the development workstation:

```
\Program Files\Common Files\Microsoft shared\CoreCon\1.0\Target\WCE400\<CPU>
```

There are multiple set of CoreCon files installed to the sub-folders under the above directory, to support different families of processor, with the processor family as the sub-folder name. Each of these CoreCon files supports a designated processor family. The VDX-6326 SBC is engineered with an x86 processor. The CoreCon files in the following folder are needed to support the application development exercises in this guide.

```
\Program Files\Common Files\Microsoft shared\CoreCon\1.0\Target\WCE400\x86\
```

To establish connectivity between the Visual Studio IDE and CE 6.0 target device, the appropriate CoreCon files need to be copied to the target device's local storage, or make available to the target device via Network share or external storage device.

To help ease the process of including the required CoreCon files to the OS design, an installable CoreCon component for CE 6.0, `CoreCon_v200_x86_WinCE600.msi`, is provided with this jump start kit. Locate and launch the `CoreCon_v200_x86_WinCE600.msi` installation file on the jump start kit CD, in the `\Software` folder, to install this component.

There are different versions of CoreCon. CoreCon version 8.0 is installed by VS2005. When VS2008 is installed, it updates the CoreCon files to version 9.0.

A development workstation with both VS2005 and 2008 installed will have CoreCon version 9.0.

CE 6.0 image built with different version of CoreCon from one development workstation will not be able to establish connectivity to another development workstation with different version of CoreCon files installed.

After installing the `CoreCon_v200_x86_WinCE600.msi` file, the CoreCon component shows up under the \Third Party\CoreCon folder on the CE 6.0 Platform Builder IDE's component catalog as **CoreCon_v200_x86**. When selected and included to the OS design, the necessary CoreCon files are included as part of the OS runtime image, generated from the OS design.

The CoreCon_v200_x86 component does not include the actual CoreCon binary files. When the CoreCon_v200_x86 component is included in the OS design, it references and adds the necessary build steps to include the CoreCon files already installed to the development workstation by the Visual Studio (2005 & 2008) installation, to the OS runtime image.

Following the steps provided in this jump start guide to develop an OS design with CoreCon_v200_x86 component selected, the OS runtime image generated from the OS design will have the same version of CoreCon installed on the development workstation to support connectivity between the target device and the development workstation's VS2005 or VS2008 IDE.

Note: In order to establish connectivity between a CE 6.0 target device and the Visual Studio IDE using CoreCon, both the target device and development workstation must have the same version of CoreCon.

For development team with separate developers assigned to work on the OS design and application development, both the OS design and application development developers' workstations must have the same version of CoreCon.

CoreCon is installed to the workstation as part of the Visual Studio 2005 and 2008 installation. Although Visual Studio 2008 is not needed to develop the OS design for CE 6.0, when Visual Studio 2008 is installed on the application developer's workstation, the OS design developer's workstation mush also has Visual Studio 2008 installed to have the same version of CoreCon included in the OS runtime image.

Visual Studio 2008 installs a newer version of CoreCon than Visual Studio 2005.

In addition to including the necessary CoreCon files with the OS runtime image, the CoreCon executable must be launched from the target device in order for the Visual Studio IDE to establish connectivity to the target device.

The AutoLaunch component is provided to automatically launch the CoreCon executable when the CE 6.0 OS starts.

AutoLaunch Component

The AutoLaunch component is provided to help make it easier to automatically launch one or more application when the CE 6.0 OS starts. When included in the OS runtime image, it can be configured to launch one or more application automatically and specify delay time to launch multiple applications in a designated sequence when the CE 6.0 OS starts.

The AutoLaunch component is provided as a self installable file, `AutoLaunch_v200_x86_WinCE600.msi`, in the `\Software` folder on the jump start CD. After installation, the AutoLaunch_v200_x86 component shows up as one of the components on the Platform Builder's component catalog, in the Third Party folder.

The AutoLaunch_v200_x86 component files are installed to the following folder by default:

```
C:\WINCE600\3rdparty\AutoLaunch_v200_x86\
```

To use this component, in addition to including the component to the OS design, add the following registry entries to launch the designated application(s), to the Project.reg file:

```
[HKEY_LOCAL_MACHINE\Startup]
    "Process0"="app1.exe <startup parameter>"      ; first app
    "Process0Delay"=dword:00001388                 ; delay 5 seconds
                                                   ; 1388 Hex = 5000 in decimal
    "Process1"="app2.exe <startup parameter>"      ; second app
    "Process1Delay"=dword:2710                     ; delay 10 seconds
    "Process2"="app3.exe <startup parameter>"      ; third app
    "Process2Delay"=dword:3A98                     ; delay 15 seconds
```

RegFlushApp Component

The RegFlushApp component contains a Win32 application with a simple task, call the RegFlushKey() function, to flush and save changes to the registry to the persistence storage.

With Hive-based registry enabled, to save changes made to the registry, the system needs to have a way to flush and save these changes to the persistence storage. There are multiple methods to accomplish this:

- One of the methods is to enable Flush-On-Close registry flushing. With this option enabled, the RegFlushKey function is called every time the RegCloseKey is called. This method impacts the system's performance. To enable Flush-On-Close registry flushing, add the following registry entries to the OS design:

  ```
  [HKEY_LOCAL_MACHINE\init\BootVars]
      "RegistryFlags"=dword:1
  ```

- Another option to flush the registry is to call the RegFlushKey() function when needed. Since the registry is flush only when it's needed, and only flush the changes to the persistence storage, this method has minimal impact to system performance.

The RegFlushApp component is provided as a self installable file in the `\Software` folder, `RegFlushApp_v100_x86_WINCE600.msi`. The RegFlushApp files are installed to the following folder by default:

```
C:\WINCE600\3rdparty\RegFlushApp\
```

Note: The RegFlushApp component is provided as a sample utility. In a production environment, it's best to incorporate the RegFlushKey() function call within the application written for the device.

Sample Project Codes

The project codes for the exercises in this jump start guide are provided. The codes are provided on the jump start CD, in the `\SampleCodes` folder.

Recommended Software Installation Sequence

It's important to install the software in their proper sequences. Here is the recommended software installation sequence, in numeric order.

1. Visual Studio 2005
2. Visual Studio 2005 SP1
   If you have the full retail or evaluation version of Windows Embedded CE 6.0 R2 or R3, The VS2005 SP1 installation file is provided on one of the DISC. Otherwise, download from following URL:

http://www.microsoft.com/downloads/details.aspx?FamilyID=bb4a75ab-e2d4-4c96-b39d-37baf6b5b1dc&DisplayLang=en

3. Visual Studio 2005 SP1 update for Vista
   If you have the full retail or evaluation version of Windows Embedded CE 6.0 R2 or R3, The VS2005 SP1 update for Vista installation file is provided on one of the DISC. Otherwise, download from the following URL:
   http://www.microsoft.com/downloads/details.aspx?FamilyID=90e2942d-3ad1-4873-a2ee-4acc0aace5b6&DisplayLang=en
   Note: If you are using Windows XP machine, skip this step.

4. Windows Embedded CE 6.0

5. Windows Embedded CE 6.0 SP1
   If you have the full retail or evaluation version of Windows Embedded CE 6.0 R2 or R3, The CE 6.0 SP1 is provided on one of the DISC. Otherwise, download from the following URL:
   http://www.microsoft.com/downloads/details.aspx?FamilyID=bf0dc0e3-8575-4860-a8e3-290adf242678&DisplayLang=en

6. Windows Embedded CE 6.0 R2
   If you have the full retail or evaluation version of Windows Embedded CE 6.0 R2 or R3, The CE 6.0 R2 update is provided on one of the DISC. Otherwise, download from the following URL:
   http://www.microsoft.com/downloads/details.aspx?FamilyID=f41fc7c1-f0f4-4fd6-9366-b61e0ab59565&DisplayLang=enhttp://www.microsoft.com/downloads/details.aspx?FamilyID=bf0dc0e3-8575-4860-a8e3-290adf242678&DisplayLang=en

7. Windows Embedded CE 6.0 R3
   If you have the full retail or evaluation version of Windows Embedded CE 6.0 R3, The CE 6.0 R3 update is provided on one of the DISC. Otherwise, download from the following URL:
   http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=bc247d88-ddb6-4d4a-a595-8eee3556fe46http://www.microsoft.com/downloads/details.aspx?FamilyID=f41fc7c1-f0f4-4fd6-9366-b61e0ab59565&DisplayLang=enhttp://www.microsoft.com/downloads/details.aspx?FamilyID=bf0dc0e3-8575-4860-a8e3-290adf242678&DisplayLang=en

8. ICOP_VDX6326_60B_BSP.msi
   This BSP is provided on the jump start kit CD, in the `\Software` folder.

9. VDX6326_WINCE600_SDK.msi
   This SDK is provided on the jump start kit CD, in the `\Software` folder.

10. CoreCon_v200_x86_WinCE600.msi
    This CoreCon component is provided on the jump start kit CD, in the `\Software` folder.

11. AutoLaunch_v200_x86_WinCE600.msi

This AutoLaunch component is provided on the jump start kit CD, in the `\Software` folder.

12. RegFlushApp_v100_x86_WinCE600.msi

    This RegFlushApp component is provided in the jump start kit CD, in the `\Software` folder.

New version and update for ICOP_VDX6326_60B_BSP.msi, VDX6326_WINCE600_SDK.msi, CoreCon_v200_x86_WinCE600.msi, AutoLaunch_v200_x86_WinCE600.msi and other Windows Embedded CE resources are available from the following URL:

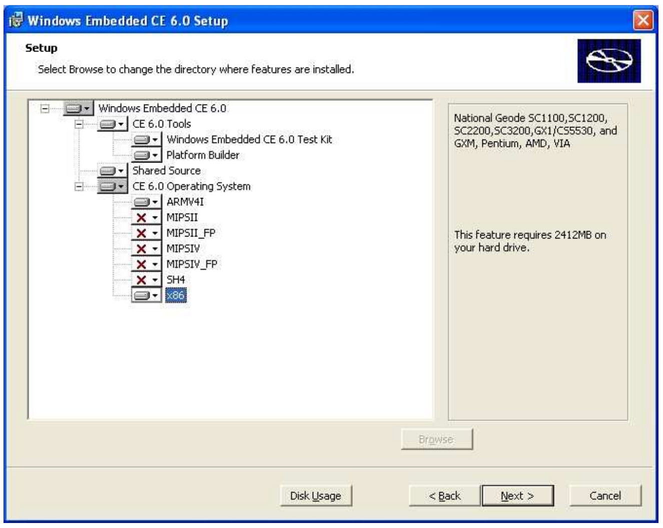
http://www.embeddedpc.net/download/

Windows Embedded CE 6.0 Installation

Since the CE 6.0 development tool, Platform Builder, is a plug-in for the VS2005 IDE, the VS2005 development tool must be installed to the develop workstation prior to installing the CE 6.0 software packages. While the CE 6.0 installation does not require VS2005 SP1 to be installed, it's required to install SDK generated by the Platform Builder. The SDK for the OS design used to generate the OS runtime image is needed to support VS2005 application development. For Windows Vista machine, VS2005 SP1 Update for Vista is also needed.

By default, the CE 6.0 installation program only selects and includes support for the ARMV4I processor during installation. The VDX-6326 SBC is built with an x86 processor. Support for the x86 processor must be selected during the installation in order to use the ICOP_VDX6326_60B BSP and build CE 6.0 OS runtime image for the VDX-6326 SBC.

When installing the CE 6.0 software, during the supported processor selection step, include support for the x86 processor, as shown in Figure 2.



Windows Embedded CE 6.0 Setup

Setup

Select Browse to change the directory where features are installed.

- Windows Embedded CE 6.0
  - CE 6.0 Tools
    - Windows Embedded CE 6.0 Test Kit
    - Platform Builder
  - Shared Source
  - CE 6.0 Operating System
    - ARMV4I
    - ✗ MIPSII
    - ✗ MIPSII_FP
    - ✗ MIPSIV
    - ✗ MIPSIV_FP
    - ✗ SH4
    - x86

National Geode SC1100,SC1200, SC2200,SC3200,GX1/CS5530, and GXM, Pentium, AMD, VIA

This feature requires 2412MB on your hard drive.

Fig. 2 - CE 6.0 installation screen, supported CPU selection

Windows Embedded CE 6.0 SP1 Installation

After installing the CE 6.0 software, install the CE 6.0 SP1 update. Depending on the version of CE 6.0 CD or DVD being used to install the software, the CE 6.0 SP1 update may be provided as part of the CD or DVD. Otherwise, download and install CE 6.0 SP1 from the following URL:

http://www.microsoft.com/downloads/details.aspx?FamilyID=bf0dc0e3-8575-4860-a8e3-290adf242678&DisplayLang=en

Windows Embedded CE 6.0 R2 Installation

After installing the CE 6.0 SP1, install the CE 6.0 R2 update. Depending on the version of CE 6.0 CD or DVD being used to install the software, the CE 6.0 R2 update may be provided as part of the CD or DVD. Otherwise, download and install CE 6.0 R2 from the following URL:

http://www.microsoft.com/downloads/details.aspx?FamilyID=f41fc7c1-f0f4-4fd6-9366-b61e0ab59565&DisplayLang=en

Windows Embedded CE 6.0 R3 Installation

After installing the CE 6.0 R2, install the CE 6.0 R3 update. Depending on the version of CE 6.0 CD or DVD being used to install the software, the CE 6.0 R3 update may be provided as part of the CD or DVD. Otherwise, download and install CE 6.0 R3 from the following URL:

http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=bc247d88-ddb6-4d4a-a595-8eee3556fe46

Board-Support-Package Installation

VS2005, CE 6.0, CE 6.0 SP1, CE 6.0 R2 and CE 6.0 R3 must be installed prior to installing the Board-Support-Package (“BSP”).

The ICOP_VDX6326_60B BSP is provided on the jump start CD, in the `\Software` folder.

```
\Software\ICOP_VDX6326_60B.msi
```

After installation, this BSP component shows up on the CE 6.0 Platform Builder component catalog as “ICOP_VDX6326_60B: x86” in the “\Third Party\BSP” folder.

> Note: x86 CPU support for CE 6.0 PLATFORM BUILDER is needed in order to use ICOP_VDX6326_60B BSP to create OS design and build CE 6.0 image for SBC.

SDK Installation

VS2005, CE 6.0, CE 6.0 SP1 and CE 6.0 R3 must be installed prior to installing the SDK.

The CE 6.0 SDK for the VDX-6326 SBC, `VDX6326_WINCE600_SDK.msi`, is provided on the jump start CD, in the `\Software` folder.

```
\Software\VDX6326_WINCE600_SDK.msi
```

CoreCon Component Installation

The CoreCon catalog component for CE 6.0 in self installable file format, `CoreCon_v200_x86_WinCE600.msi`, is provided with the jump start kit CD. To install, locate and launch this installation file on the jump start CD, in the `\Software` directory.

```
\Software\CoreCon_v200_x86.msi
```

After installation, this component shows up on the CE 6.0 Platform Builder component catalog as “CoreCon_v200_x86” in the “\Third Party\CoreCon” folder.

AutoLaunch Component Installation

The AutoLaunch catalog component for CE 6.0 in self installable file format, `AutoLaunch_v200_x86_WinCE600.msi`, is provided with the jump start kit CD. To install, locate and launch this installation file on the jump start CD, in the `\Software` directory.

```
\Software\AutoLaunch_v200_x86.msi
```

After installation, this component shows up on the CE 6.0 Platform Builder component catalog as “AutoLaunch_v200_x86” in the “\Third Party\AutoLaunch” folder.

RegFlushApp Component Installation

The RegFlushApp catalog component for CE 6.0 in self installable file format, `RegFlushApp_v100_x86_WinCE600.msi`, is provided with the jump start kit CD. To install, locate and launch this installation file on the jump start CD, in the `\Software` directory.

```
\Software\RegFlushApp_v100_x86.msi
```

After installation, this component shows up on the CE 6.0 Platform Builder component catalog as “RegFlushApp” in the “\Third Party\RegFlushApp” folder.

Part 4 – Common Terminology

To minimize the need to write long description, it's a common practice for developer in different industry to use abbreviated key word. To the new comer, without knowing the terminology used and what the abbreviated key words represent can cause quite a bit of confusion.

For the CE 6.0 development environment and the materials written for this jump start guide, here is a list of the abbreviated key words, terminologies and the associated description for them.

Abbreviation/ key word	**Description and Representation**
CE 6.0	"Windows Embedded CE 6.0", "Windows Embedded CE 6.0 R2" and "Windows Embedded CE 6.0 R3"
VS2005	Visual Studio 2005
VS2008	Visual Studio 2008
VB2005	Visual Basic 2005
VB2008	Visual Basic 2008
IDE	Integrated Development Environment
BSP	Board Support Package. It's a set of software package that includes the OEM adaptation layer code, device drivers and configuration files for a specific hardware platform.
OAL	OEM Adaptation Layer
Platform Builder	Windows Embedded CE 6.0 Platform Builder, the development tool to develop OS design and generate OS runtime image for Windows Embedded CE.
Component	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.
Catalog	The component catalog contains all of the components installed to the Platform Builder, or provided by default by the Platform Builder. The component catalog lists all of the available components from the Platform Builder and provides the interface to add components to the OS design, and view which components are currently added to the OS design.
OS design	OS design is a Platform Builder project, containing components that make up the OS runtime image. Windows Embedded CE OS runtime image is generated from an OS design.
Hardware Platform	Refer to the physical hardware for running the CE 6.0 OS and application.
Target Device	Refer to the hardware platform used in the Windows Embedded CE development environment.
OS Runtime Image	The binary file generated from the OS design project.
SBC	Single-Board-Computer
Release Directory or Build Release Directory	Referring to the directory where all files related to the OS design projects are placed by the build tools and compile the OS runtime image.

Within the CE 6.0 development environment, environment variables are used to represent long folder name and configure system components. Knowing what these environment variables represent help make it easier to understand project codes for CE 6.0. Here are links to MSDN pages with more information about these variables:

```
http://msdn.microsoft.com/en-us/library/aa908877.aspx
http://msdn.microsoft.com/en-us/library/aa909549.aspx
http://msdn.microsoft.com/en-us/library/aa909715.aspx
http://msdn.microsoft.com/en-us/library/aa909547.aspx
```

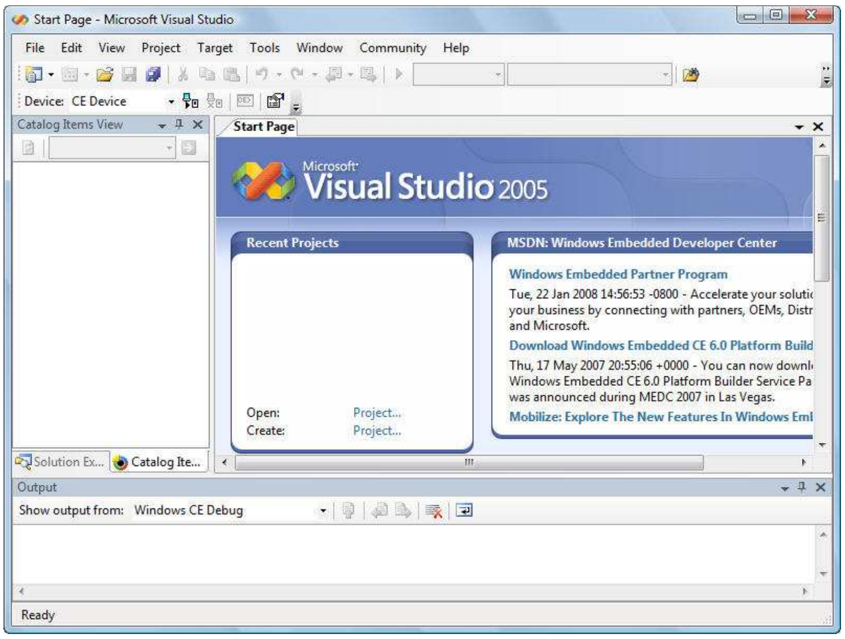
Part 5 – Develop, Configure & Customize OS Design

This section will guide you through the process to develop and configure an OS design using the New OS design wizard within Platform Builder. After the initial OS design project is created, the OS design can be customized further by removing components from the project, add additional CE 6.0 components from the catalog and make changes to the configuration and registry files to control how the final CE 6.0 OS runtime image behave.

Visual Studio 2005 IDE

CE 6.0 Platform Builder is a plug-in to the VS2005 IDE. To use Platform Builder, you need to launch VS2005 and access Platform Builder from the VS2005 IDE.

After launching, the VS2005 IDE screen is shown, similar to the screen as shown in Figure 3.



Start Page - Microsoft Visual Studio
File Edit View Project Target Tools Window Community Help
Device: CE Device
Catalog Items View
Start Page
Microsoft Visual Studio 2005
Recent Projects
MSDN: Windows Embedded Developer Center
Open: Project...
Create: Project...
Solution Ex...
Catalog Ite...
Output
Show output from: Windows CE Debug
Ready

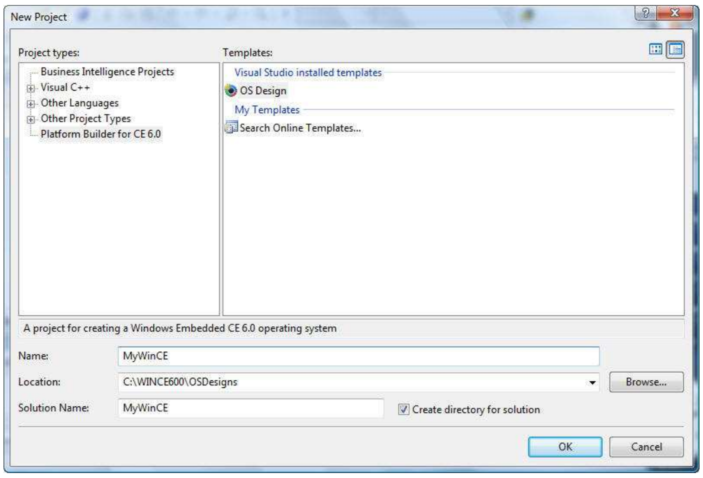
Windows Embedded Partner Program
Tue, 22 Jan 2008 14:56:53 -0800 - Accelerate your solutio your business by connecting with partners, OEMs, Distr and Microsoft.
Download Windows Embedded CE 6.0 Platform Build
Thu, 17 May 2007 20:55:06 +0000 - You can now downl Windows Embedded CE 6.0 Platform Builder Service Pa was announced during MEDC 2007 in Las Vegas.
Mobilize: Explore The New Features In Windows Em

Fig. 3 - VS2005 IDE

Note: To make it easier for capturing the Visual Studio IDE screen and create documentation for this jump start guide, the Visual Studio IDE is resized to 800x600 and may have different looks and feel from the VS2005 IDE, launch in full screen mode with 1024x768 and higher display resolution.

The VS2005 IDE provides a common environment to develop Visual Basic, Visual C++, Visual C#, Visual J# and Platform Builder solutions. Depending on the development preference selected during installation, your VS2005 screen may be little different from the above.

The VS2005 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 VS2005, the VS2005 IDE provides wizards and templates to help create the initial set of files for the project. Platform Builder for CE 6.0 is one of the available project types. From the VS2005 IDE, select "**File | New | Project …**" to bring up the new project screen, as shown in Figure 4.


New Project

Project types:
- Business Intelligence Projects
- Visual C++
- Other Languages
- Other Project Types
- Platform Builder for CE 6.0

Templates:
Visual Studio installed templates
- OS Design

My Templates
- Search Online Templates...

A project for creating a Windows Embedded CE 6.0 operating system

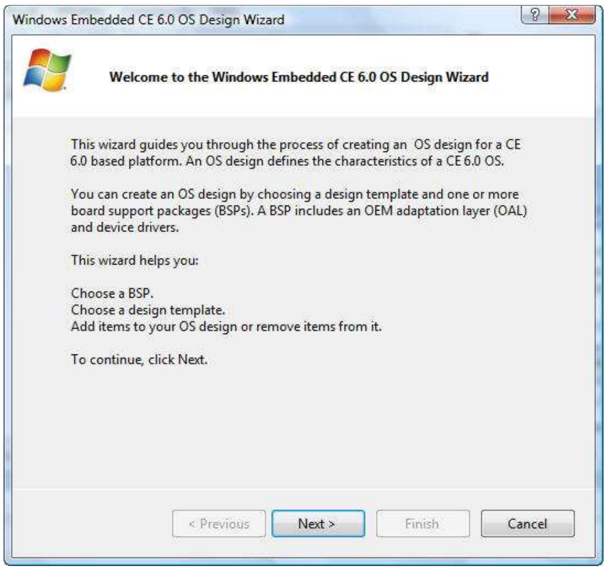
Name: MyWinCE
Location: C:\WINCE600\OSDesigns
Solution Name: MyWinCE
[✓] Create directory for solution

Fig. 4 - VS2005 New Project – CE 6.0 OS design

- From the New Project screen’s left pane, click to highlight the **Platform Builder for CE 6.0** option.
- From the right pane, click to high light the **OS design** option.
- Enter **MyWinCE** as the name of the OS design project.
- Make sure the **Create directory for solution** check box is checked.
- Click on the **OK** button to continue.

Windows Embedded CE 6.0 OS Design Wizard

When a new CE 6.0 OS design project is selected, the VS2005 IDE launches the Windows Embedded CE 6.0 OS design wizard and guides through the steps to help configure the initial OS design project, as shown in Figure 5.


Windows Embedded CE 6.0 OS Design Wizard

Welcome to the Windows Embedded CE 6.0 OS Design Wizard

This wizard guides you through the process of creating an OS design for a CE 6.0 based platform. An OS design defines the characteristics of a CE 6.0 OS.

You can create an OS design by choosing a design template and one or more board support packages (BSPs). A BSP includes an OEM adaptation layer (OAL) and device drivers.

This wizard helps you:

Choose a BSP.
Choose a design template.
Add items to your OS design or remove items from it.

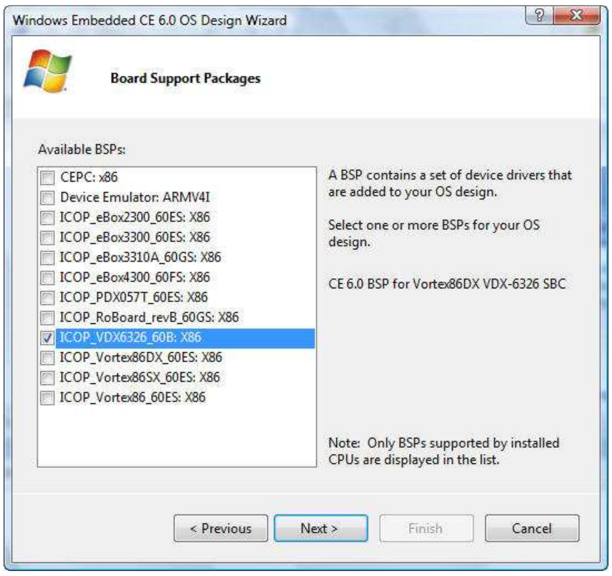
To continue, click Next.

Fig. 5 - Windows Embedded CE 6.0 OS design wizard

- Click on the **Next** button to continue and bring up the BSP selection screen.

OS Design Wizard – Board Support Packages (BSPs)

In the BSP selection step, the OS design wizard provides the option to select one or more BSP for the new OS design project, from the list of available BSP. All of the installed BSPs, including BSPs from third party companies are listed, as shown in Figure 6.


Windows Embedded CE 6.0 OS Design Wizard

Board Support Packages

Available BSPs:

- [ ] CEPC: x86
- [ ] Device Emulator: ARMV4I
- [ ] ICOP_eBox2300_60ES: X86
- [ ] ICOP_eBox3300_60ES: X86
- [ ] ICOP_eBox3310A_60GS: X86
- [ ] ICOP_eBox4300_60FS: X86
- [ ] ICOP_PDX057T_60ES: X86
- [ ] ICOP_RoBoard_revB_60GS: X86
- [x] ICOP_VDX6326_60B: X86
- [ ] ICOP_Vortex86DX_60ES: X86
- [ ] ICOP_Vortex86SX_60ES: X86
- [ ] ICOP_Vortex86_60ES: X86

A BSP contains a set of device drivers that are added to your OS design.

Select one or more BSPs for your OS design.

CE 6.0 BSP for Vortex86DX VDX-6326 SBC

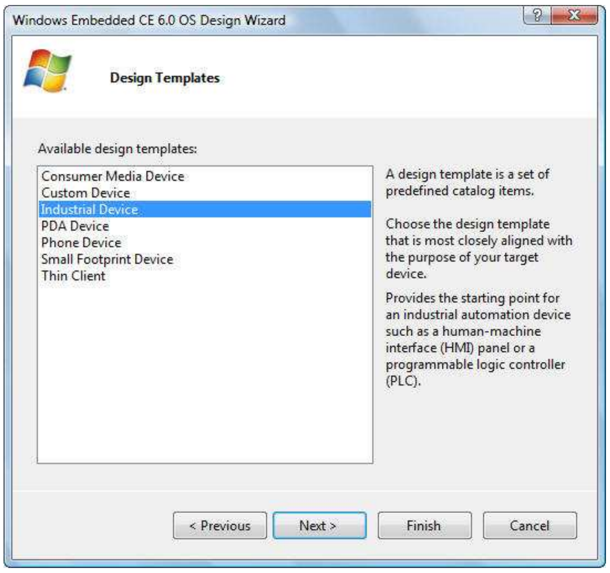
Note: Only BSPs supported by installed CPUs are displayed in the list.

Fig. 6 - OS design wizard – Select BSP

- From the Available BSPs pane, select **ICOP_VDX6326_60B: x86** BSP.
- Click on the **Next** button to continue, and bring up the design template screen.

OS Design Wizard – Design Templates

In the design templates selection step, the OS design wizard provides a list containing multiple design templates to choose from, as shown in Figure 7.


Windows Embedded CE 6.0 OS Design Wizard

Design Templates

Available design templates:

Consumer Media Device
Custom Device
Industrial Device
PDA Device
Phone Device
Small Footprint Device
Thin Client

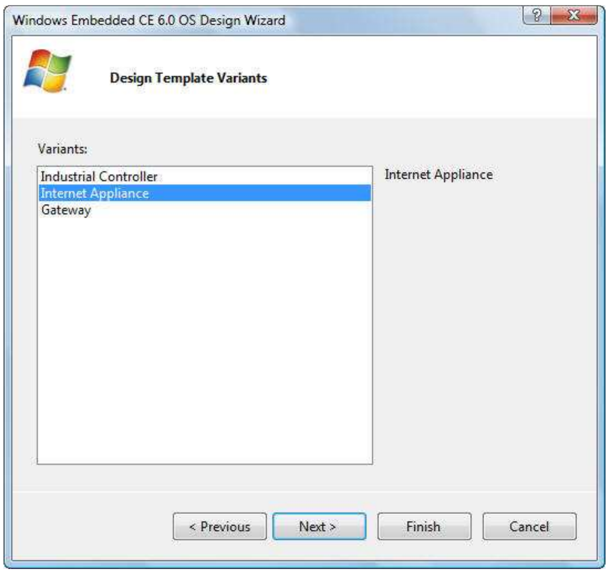
A design template is a set of predefined catalog items.

Choose the design template that is most closely aligned with the purpose of your target device.

Provides the starting point for an industrial automation device such as a human-machine interface (HMI) panel or a programmable logic controller (PLC).

Fig. 7 - OS design wizard – Design Templates

- Click to high light and select the **Industrial Device** option.
- Click on the **Next** button to continue and bring up the **Design Template Variants** selection screen, as shown in Figure 8.



Variants:
Industrial Controller
Internet Appliance
Gateway

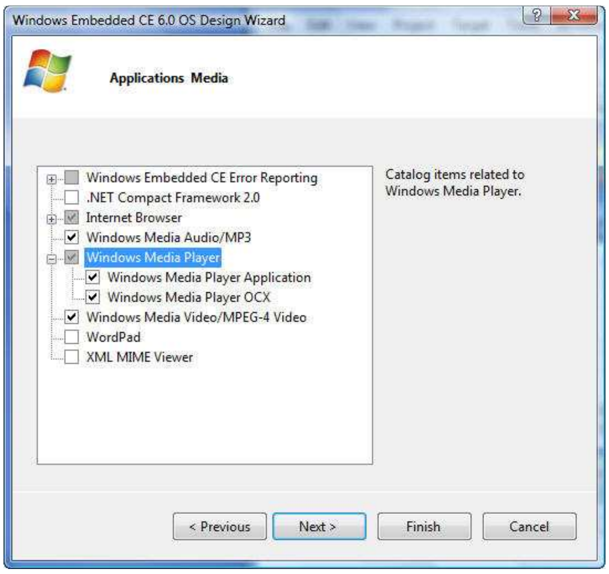
Internet Appliance

Fig. 8 - OS design wizard – Design Template Variants

- Click to high light and select the **Internet Appliance** option.
- Click on the **Next** button to continue, and bring up the Application & Media screen.

OS Design Wizard – Applications & Media

In the applications & media selection step, the OS design wizard provides the options to select .NET Compact Framework to support managed code applications, Internet Explorer, Windows Media components, and etc., as shown in Figure 9.



- [ ] Windows Embedded CE Error Reporting
- [ ] .NET Compact Framework 2.0
- [✓] Internet Browser
- [✓] Windows Media Audio/MP3
- [✓] Windows Media Player
  - [✓] Windows Media Player Application
  - [✓] Windows Media Player OCX
- [✓] Windows Media Video/MPEG-4 Video
- [ ] WordPad
- [ ] XML MIME Viewer

Catalog items related to Windows Media Player.

Fig. 9 - OS design wizard – Applications & Media

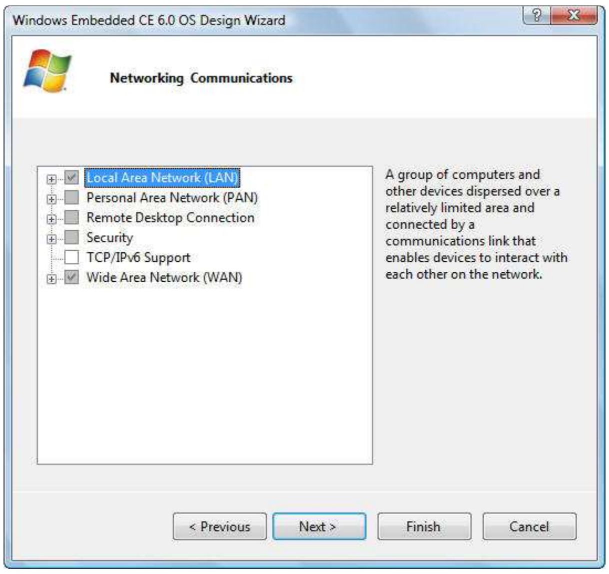
In this step, perform the following:

- Uncheck the .NET Compact Framework 2.0 to remove from the selection.
  (In a later step, we will include a newer version, .NET Compact Framework 3.5, to support managed code application.)
- Select Internet Explorer 6.0.
- Select Windows Media Audio/MP3.
- Select Windows Media Player Application.
- Select Windows Media Player OCX.
- Select Windows Media Video/MPEG-4 Video.
- Click on the **Next** button to continue, and bring up the Networking & Communication screen.

Note: .NET Compact Framework is needed to support managed code application. In this step, the .NET Compact Framework 2.0 library is excluded. In the later step, we will add the newer version, .NET Compact Framework 3.5, to the OS design.

OS Design Wizard – Networking & Communications

In the networking & communications step, the OS design wizard provides the options to select communication, networking and security components, as shown in Figure 10.


Windows Embedded CE 6.0 OS Design Wizard

Networking Communications

- [✓] Local Area Network (LAN)
- [ ] Personal Area Network (PAN)
- [ ] Remote Desktop Connection
- [ ] Security
- [ ] TCP/IPv6 Support
- [✓] Wide Area Network (WAN)

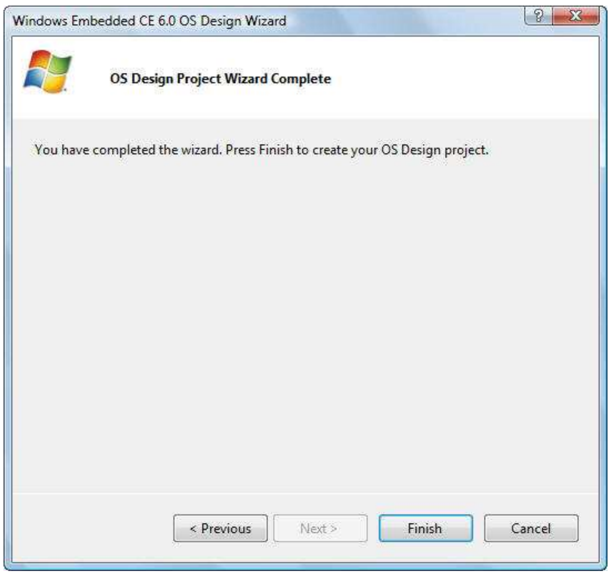
A group of computers and other devices dispersed over a relatively limited area and connected by a communications link that enables devices to interact with each other on the network.

Fig. 10 - OS design wizard – Networking & Communications

We will use the default settings for networking and communications. Click on the **Next** button to continue.

OS Design Wizard – Complete

At this point, the OS design wizard collected the necessary OS design parameters to configure the initial OS design project based on the selected OS design template and components.


Windows Embedded CE 6.0 OS Design Wizard

OS Design Project Wizard Complete

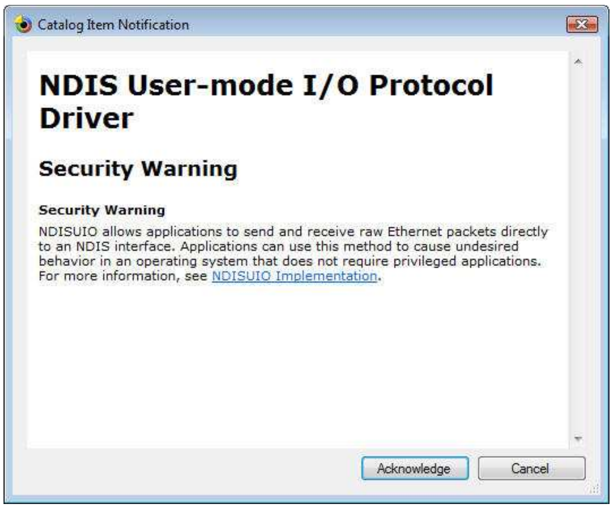
You have completed the wizard. Press Finish to create your OS Design project.

Fig. 11 - OS design wizard – Completed

- Click on the **Finish** button to complete the OS design wizard step.

Catalog Item Notification

At the completion of OS design wizard, a security warning is raised to warn one or more of the selected components included in the OS design may pose security risk, as shown in Figure 12.


Catalog Item Notification

NDIS User-mode I/O Protocol Driver

Security Warning

Security Warning

NDISUIO allows applications to send and receive raw Ethernet packets directly to an NDIS interface. Applications can use this method to cause undesired behavior in an operating system that does not require privileged applications. For more information, see NDISUIO Implementation.

Fig. 12 - Catalog Item Notification – Security Warning

Click on the **Acknowledge** button to close the warning screen.

At the completion of the OS design wizard, Platform Builder generates the initial OS design project files and pulls in all required files associate with the OS design template and selected components.

Part 6 – Customize and Build the OS Design

At this point, with help from the OS design wizard, the initial MyWinCE OS design project is created using the Internet Appliance design template along with the selected components and ICOP_VDX6326_60B BSP.

The following project folder and sub-folders are created for the project, under the main CE 6.0 Platform Builder's OS design directory, \WINCE600\OSDesigns.

- \WINCE600\OSDesigns\MyWinCE\

  This is the folder for the MyWinCE Solution

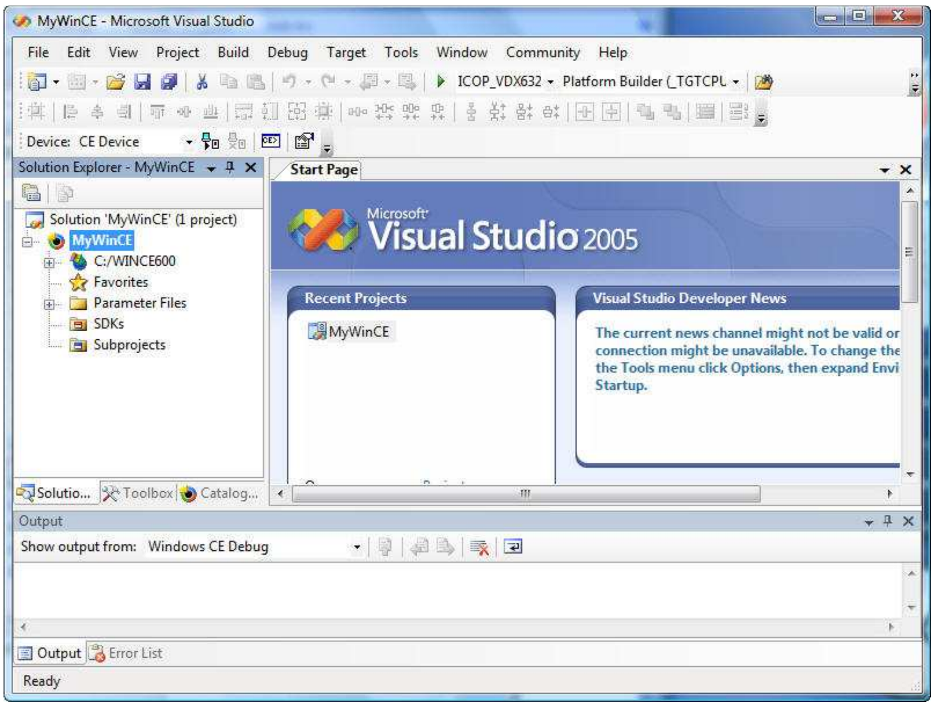
  VS2005 supports different project types. A VS2005 solution provides a centralized work space to keep different project types supporting the same solution in one location.

  For example, the MyWinCE solution may include the "MyWinCE OS design", a "Visual Basic managed code application", a "Visual C# managed code application" and a "Visual C++ native code application".

- C:\WINCE600\OSDesigns\ MyWinCE\ MyWinCE\

  This is the folder for the MyWinCE CE 6.0 Platform Builder project, an OS design.

Your VS2005 IDE should look similar to the screen as shown in Figure 13.


MyWinCE - Microsoft Visual Studio
File Edit View Project Build Debug Target Tools Window Community Help
ICOP_VDX632 Platform Builder (_TGTCPL
Device: CE Device
Solution Explorer - MyWinCE
Solution 'MyWinCE' (1 project)
MyWinCE
C:/WINCE600
Favorites
Parameter Files
SDKs
Subprojects
Start Page
Microsoft Visual Studio 2005
Recent Projects
MyWinCE
Visual Studio Developer News
Solutio... Toolbox Catalog...
Output
Show output from: Windows CE Debug
Output Error List
Ready

The current news channel might not be valid or connection might be unavailable. To change the the Tools menu click Options, then expand Envi Startup.

Fig. 13 - VS2005 IDE after OS design wizard

Customize the OS Design – Additional Catalog Components

The OS design can be further customized by the following:

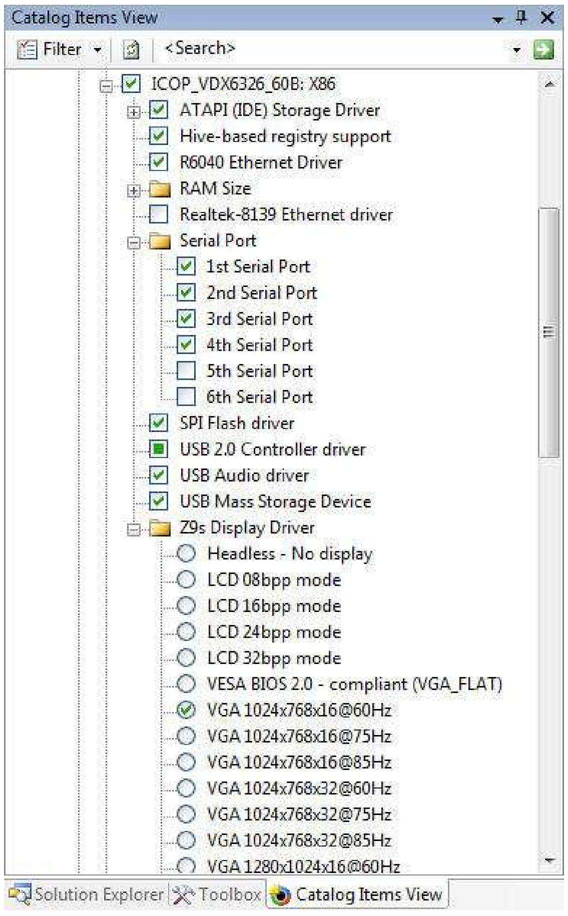
- Add component(s) to the OS design.
- Remove component(s) from the OS design.
- Add application and library as subproject to the OS design.
- Modify system configuration and registry files.

The Catalog Item View window, on the Platform Builder IDE, lists all of the available CE 6.0 components, including applications, library, drivers, utilities & 3rd party components available to be added to the OS design.

On the Catalog Item View windows, the green check mark to the left of a component indicates the component is selected to be part of the OS design. The solid green square to the left of a component indicates the component is a dependency component to another component that is selected to include to the OS design.

Work through the following steps to further customize the MyWinCE OS design project:

- From the VS2005 IDE, click on the Catalog Items View tab, expand the **Third Party\BSP\ICOP_VDX6326_60B: x86** folder, as shown in Figure 14.



- ✓ ICOP_VDX6326_60B: X86
  - ✓ ATAPI (IDE) Storage Driver
  - ✓ Hive-based registry support
  - ✓ R6040 Ethernet Driver
  - RAM Size
  - ☐ Realtek-8139 Ethernet driver
  - Serial Port
    - ✓ 1st Serial Port
    - ✓ 2nd Serial Port
    - ✓ 3rd Serial Port
    - ✓ 4th Serial Port
    - ☐ 5th Serial Port
    - ☐ 6th Serial Port
  - ✓ SPI Flash driver
  - ■ USB 2.0 Controller driver
  - ✓ USB Audio driver
  - ✓ USB Mass Storage Device
  - Z9s Display Driver
    - ○ Headless - No display
    - ○ LCD 08bpp mode
    - ○ LCD 16bpp mode
    - ○ LCD 24bpp mode
    - ○ LCD 32bpp mode
    - ○ VESA BIOS 2.0 - compliant (VGA_FLAT)
    - ✓ VGA 1024x768x16@60Hz
    - ○ VGA 1024x768x16@75Hz
    - ○ VGA 1024x768x16@85Hz
    - ○ VGA 1024x768x32@60Hz
    - ○ VGA 1024x768x32@75Hz
    - ○ VGA 1024x768x32@85Hz
    - ○ VGA 1280x1024x16@60Hz

Fig. 14 - Component Catalog

Note: The ICOP_VDX6326_60B BSP is created to support the VDX-6326 SBC. To help make the BSP easier to use, some of this BSP's components enable one or more environment variables to include device driver and support library provided by Platform Builder to the OS design.

- Check and verify the following BSP components are selected and included to the OS design:
  - **ATAPI (IDE) Storage Driver**

    Note: The VDX-6326 SBC's ATAPI storage driver set the SYSGEN_ATAPI variable to include the ATAPI storage driver and SYSGEN_FATFS variable to include FAT file system support, provided by the Platform Builder.
  - **Hive-based registry support**

    Note: The Hive-based registry component is needed to save registry settings to non volatile flash storage when the SBC power off.
  - **R6040 Ethernet driver**

    Note: The VDX-6326 SBC has 3 10/100 Mbps Ethernet ports onboard. One of the Ethernet port use the R6040 controller which is built-in to the Vortex86DX SoC. The other two Ethernet port use the Realtek-8100 controller, which is supported by the Realtek-8139 Ethernet driver.

    For the purpose of this exercise, we only need one Ethernet interface and include the R6040 device driver to the OS design project.
  - **256MB RAM**

    Note: The VDX-6326 SBC is built with 256MB of system memory. This component set the IMGRAM256 environment variable and configure the OS runtime image to use the 256MB of available system memory.
  - **1st serial port**

    Note: This component set the BSP_SERIAL environment variable to include serial port driver to the OS runtime to support COM1, the first serial port.
  - **2nd serial port**

    Note: This component set the BSP_SERIAL2 environment variable to include serial port driver to the OS runtime to support COM2, the second serial port.
  - **3rd serial port**

    Note: This component set the BSP_SERIAL3 environment variable to include serial port driver to the OS runtime to support COM3, the third serial port.
  - **4th serial port**

    Note: This component set the BSP_SERIAL4 environment variable to include serial port driver to the OS runtime to support COM4, the fourth serial port.

By default, the VDX-6326 SBC's 4 serial ports are configured as follow:

- COM1: IRQ4, 3F8h
- COM2: IRQ3, 2F8h
- COM3: IRQ10, 3E8h
- COM4: IRQ11, 2E8h

- **SPI Flash driver**

  Note: The SPI Flash driver supports the 4MB SPI flash built-in to the VDX-6326 SBC.

- **USB 2.0 Controller driver**

  Note: This component set the SYSGEN_USB, BSP_USB_OHCI, BSP_USB_UHCI and BSP_USB_EHCI environment variables to include the USB 2.0 driver.

- **USB Audio driver**

  Note: This component set the BSP_VORTEX86DX_USB_AUDIO, SYSGEN_USB, BSP_USB_OHCI, BSP_USB_UHCI and BSP_USB_EHCI environment variables to include the USB 2.0 driver and audio driver.

- **USB Mass Storage Device**

  Note: This component set the SYSGEN_USB_STORAGE environment variable to include the USB storage class driver to the OS runtime image to support external USB storage.

- **VGA 1024x768x16 @ 60Hz**

  Note: You can select a different display setting supported by the display monitor you are working with.

The ICOP_VDX6326_60B BSP provides multiple selections to configure system memory and video display resolution. The VDX-6326 SBC is built with 256MB system memory (RAM). When the 256MB RAM component is selected, the IMGRAM256 variable is set to configure the OS design to generate an OS runtime image to support 256MB of RAM.

The video display settings can be configured to a setting other than the selected 1024x768x16 @ 60Hz, supported by the display monitor you are using.

• Expand **Core OS\CEBASE** folder, locate and include the following components to the OS design.

  - **\Applications-End User\CAB File Installer/Uninstaller**

    Note: This component provides application installation & removal support. It's needed to support application deployment using VS2005 and VS2008.

• .NET Compact Framework components are needed to support managed code application. During the OS design wizard steps, .NET Compact Framework 2.0 components were removed from the selection. In this step, include the following newer version, .NET Compact Framework 3.5, to the OS design:

  - **.NET Compact Framework 3.5**
  - **OS Dependencies for .NET Compact Framework 3.5**

    Note: If you are using an earlier release of the CE 6.0, .NET Compact Framework 3.5 is available as part of the January 2008 QFE updates for CE 6.0.

    Otherwise, you can include the .NET Compact Framework 2.0 components to the OS design

- Expand **Third Party\CoreCon** folder and select the **CoreCon_v200_x86** component. By selecting this component, the CoreCon files are included to the OS runtime image, needed to establish connectivity between the CE 6.0 device and VS2005 (or VS2008) development workstation.

  To establish connectivity between the CE 6.0 device and VS2005 development workstation, the CoreCon component needs to be launched from the CE 6.0 device when CE 6.0 starts. To accomplish this, we use the AutoLaunch utility to launch the CoreCon component when CE 6.0 starts. The following step adds the AutoLaunch component to the OS design.

- From the **Third Party\AutoLaunch** folder, select and include the **AutoLaunch_v200_x86** component to the OS design.

  With appropriate registry entries added to the OS design, the AutoLaunch utility can be configured to launch one or more application automatically when the CE 6.0 OS starts.

```
[HKEY_LOCAL_MACHINE\Startup]
        “Process1”=” ConmanClient2.exe”         ; Launch CoreCon
        “Process1Delay”=dword:00001388          ; delay 5s 1388(Hex) = 5000(decimal)
        “Process2”=”Cerdisp.exe -c”             ; Launch Remote Display application
        “Process2Delay”=dword:00002710          ; delay 5s 2710(Hex) = 10000(decimal)
```

- From the **Third Party\RegFlushApp** folder, select and include the **RegFlushApp** component to the OS design. When this component is selected, the RegFlushApp.exe application is included in the OS design and built as part of the runtime image.

  The RegFlushApp application works in conjunction with Hive-based registry. With Hive-based registry enabled, when changes are made to the registry, the system needs a mechanism to save these changes. The RegFlushApp, when launched, call the RegFlushKey() function to flush and save the registry.

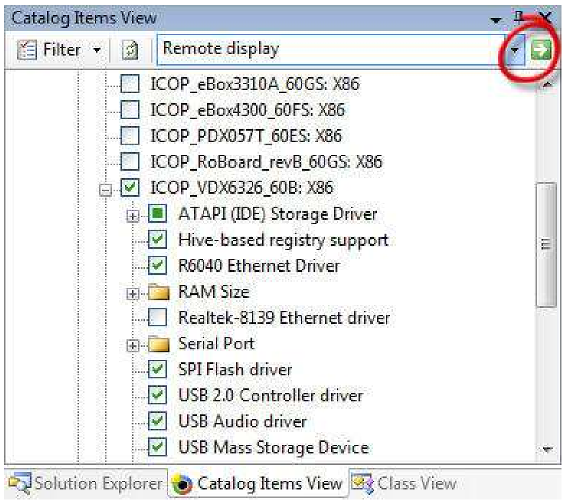
  The RegFlushApp is accessible from the Start menu on the CE 6.0 desktop, as follow:

  Start | Programs | RegFlushApp

Customize the OS Design – Locate Component by Search

The Platform Builder IDE also provides a search function to locate component from the catalog by searching the catalog using partial key word associated with the component. In this section, we will demonstrate how to locate a component from the catalog using the search function.

- From the Catalog Item View tab, enter “Remote display” in the search text box on the top right corner, as shown in Figure 15, and click on the green arrow to the right of the search text box to search for the component.



Catalog Items View

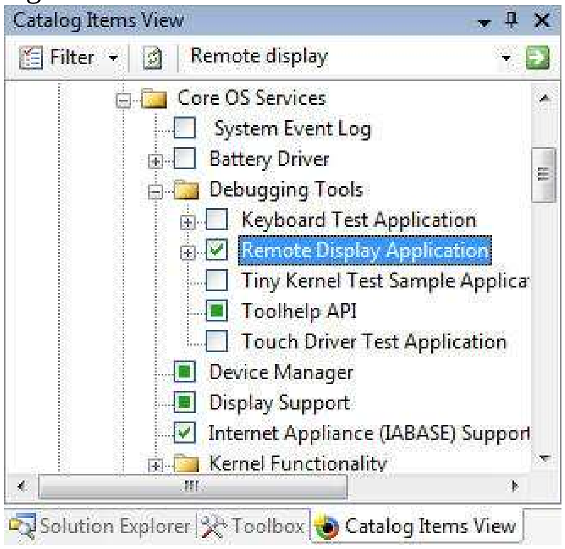
Filter | Remote display

- ☐ ICOP_eBox3310A_60GS: X86
- ☐ ICOP_eBox4300_60FS: X86
- ☐ ICOP_PDX057T_60ES: X86
- ☐ ICOP_RoBoard_revB_60GS: X86
- ✓ ICOP_VDX6326_60B: X86
  - ▣ ATAPI (IDE) Storage Driver
  - ✓ Hive-based registry support
  - ✓ R6040 Ethernet Driver
  - RAM Size
  - ☐ Realtek-8139 Ethernet driver
  - Serial Port
  - ✓ SPI Flash driver
  - ✓ USB 2.0 Controller driver
  - ✓ USB Audio driver
  - ✓ USB Mass Storage Device

Solution Explorer | Catalog Items View | Class View

Fig. 15 - Locate component using the search feature

- The search engine locates and highlights the Remote Display Application, as shown in Figure 16.



Catalog Items View

Filter | Remote display

- Core OS Services
  - ☐ System Event Log
  - ☐ Battery Driver
  - Debugging Tools
    - ☐ Keyboard Test Application
    - ✓ Remote Display Application
    - ☐ Tiny Kernel Test Sample Applica
    - ▣ Toolhelp API
    - ☐ Touch Driver Test Application
  - ▣ Device Manager
  - ▣ Display Support
  - ✓ Internet Appliance (IABASE) Suppor
  - Kernel Functionality

Solution Explorer | Toolbox | Catalog Items View

Fig. 16 - Locate component using the search feature

- Select and include the **Remote Display Application** component to the OS design.

The Remote Display Application provides the function to access the CE 6.0 desktop remotely, similar to the Remote Desktop feature available with the Windows XP Professional OS.

You can configure the AutoLaunch utility to launch the Remote Display application (Cerdisp.exe) when CE 6.0 OS starts.

To access the CE 6.0 desktop, with the Remote Display Application launched on the CE 6.0 target device. From the VS2005 development workstation, launch the CERHOST.EXE executable from the following folder:

\WINCE600\Public\Common\Oak\Bin

Customize the OS Design – Configuration Manager

Using the configuration manager, the OS design can be configured to generate an OS runtime image in debug or release mode. A debug mode image will provide more detailed debug messages when the compiled OS image loads and executes applications and modules. The size of the debug image is generally about 50% larger, comparing to the release mode image built from the same OS design. A release mode image with KITL enabled provides sufficient debug information for most of the general application development environment. For the purpose of this guide, we will configure the OS design to generate a release mode image.

In the later section, we will work through the steps to show how to debug a CE 6.0 OS runtime image with KITL enabled, using remote tools.

From VS2005 IDE, select **Build | Configuration Manager…** to bring up the Configuration Manager screen, as shown in Figure 17.

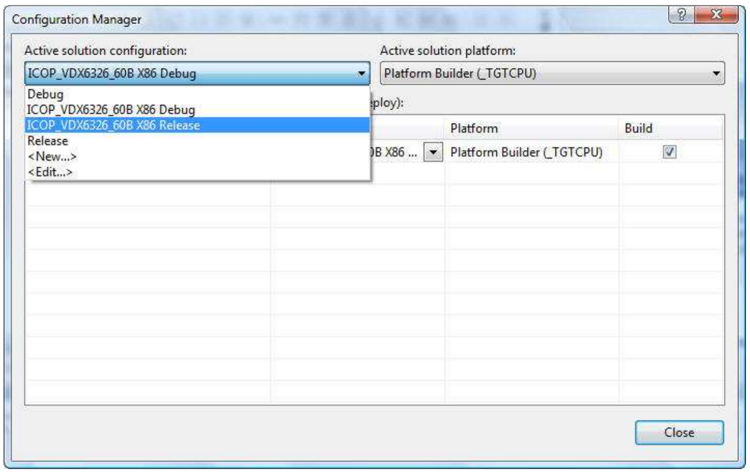

Configuration Manager
Active solution configuration:
ICOP_VDX6326_60B X86 Debug
Debug
ICOP_VDX6326_60B X86 Debug
ICOP_VDX6326_60B X86 Release
Release
<New...>
<Edit...>
Active solution platform:
Platform Builder (_TGTCPU)
ploy):
Platform
Build
)B X86 ...
Platform Builder (_TGTCPU)
Close

Fig. 17 - Configuration Manager

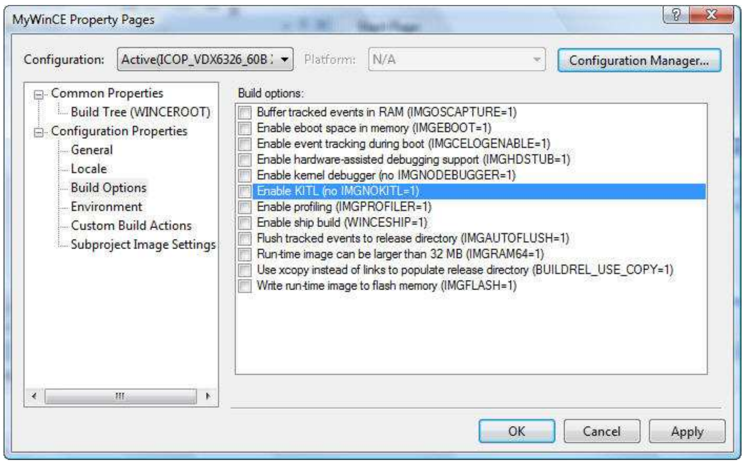
- From the **Active solution configuration** selection options on the **Configuration Manager** screen, select **ICOP_VDX6326_60B x86 Release** and click the Close button to set the OS design to generate a release mode image.

A Debug image provides more detailed system status and activities information during startup of the OS and when application modules are executing. To generate a debug image, repeat this step and select ICOP_VDX6326_60B x86 Debug option instead, and continue to finish building a debug image.

Customize the OS Design – Build Options

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

From VS2005 IDE, select **Project | MyWinCE Properties…** to bring up the MyWinCE Property Pages screen, as shown in Figure 18.



MyWinCE Property Pages

Configuration: Active(ICOP_VDX6326_60B Platform: N/A

Common Properties
- Build Tree (WINCEROOT)

Configuration Properties
- General
- Locale
- Build Options
- Environment
- Custom Build Actions
- Subproject Image Settings

Build options:
- [ ] Buffer tracked events in RAM (IMGOSCAPTURE=1)
- [ ] Enable eboot space in memory (IMGEBOOT=1)
- [ ] Enable event tracking during boot (IMGCELOGENABLE=1)
- [ ] Enable hardware-assisted debugging support (IMGHDSTUB=1)
- [ ] Enable kernel debugger (no IMGNODEBUGGER=1)
- [ ] Enable KITL (no IMGNOKITL=1)
- [ ] Enable profiling (IMGPROFILER=1)
- [ ] Enable ship build (WINCESHIP=1)
- [ ] Flush tracked events to release directory (IMGAUTOFLUSH=1)
- [ ] Run-time image can be larger than 32 MB (IMGRAM64=1)
- [ ] Use xcopy instead of links to populate release directory (BUILDREL_USE_COPY=1)
- [ ] Write run-time image to flash memory (IMGFLASH=1)

Fig. 18 - OS design Property – Build Options

- Click to expand the **Configuration Properties** node on the left side of the screen.
- Click to high light the **Build Options** node, a list of Build Options with check boxes are shown on the Build options pane on the right side. The following two build options are selected by default:
  - **Enable eboot space in memory (IMGEBOOT=1)**
  - **Enable KITL (no IMGNOKITL=1)**
- For this exercise, none of the build option is needed. Uncheck any selected build option.

Note: The KITL connection is used to establish connectivity between the development workstation and the SBC when the runtime image is downloaded from the workstation to the SBC.

When deploying a CE 6.0 OS runtime image generated with KITL enabled on to the SBC's local flash storage, the system will search for the KITL connection during start up and fails to boot.

In a later step, we will generate an image with KITL enabled, download the image to the SBC and use remote tool to debug the image running on the SBC.

Click on the Apply button follow by the OK button to close the MyWinCE Property Pages screen.

Customize the OS Design – The Registry

The registry plays a key role in controlling how the CE 6.0 OS runtime behaves, loads driver, application and more. Improper registry entries can cause series problem and can prevent the system from loading.

In the earlier steps, we included the CoreCon_v200_x86 and AutoLaunch_v200_x86 components to the OS design. The CoreCon component is needed to establish connectivity between the Visual Studio (2005 & 2008) IDE and the CE 6.0 target device, the SBC. To accomplish this, the CoreCon executable, ConmanClient2.exe, must be launched from the target device when the CE 6.0 OS starts. The AutoLaunch component is added to handle the tasks of launching the CoreCon executable automatically when the CE 6.0 OS start. We need to configure proper registry entries for the AutoLaunch utility to perform the tasks and launch the CoreCon executable.

Work through the following steps to enter the necessary registry entries to the OS design:

- From VS2005 IDE, click on the Solution Explorer tab.
- Expand the **\Parameter Files** folder.
- Expand the **\ICOP_VDX6326_60B: X86** folder.
- Double click on **Project.reg** to open this registry file in the code editor window.
- On the code editor window's lower left, click on the Source icon to view the **Project.reg** registry file in source codes format.
- Scroll to the end of the Project.reg file and add the following entries, to launch the CoreCon component when CE 6.0 starts:

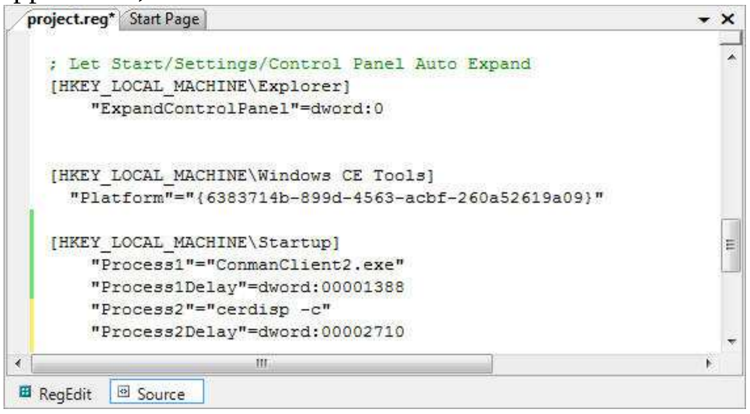
```
[HKEY_LOCAL_MACHINE\Startup]
                "Process1"="ConmanClient2.exe"
                "Process1Delay"=dword:00001388       ; delay 5 seconds
```

In the earlier steps, we also included the Remote Display application to the OS design. To utilize the Remote Display application, we need to add registry entries for the AutoLaunch utility to launch the Remote Display application when the CE 6.0 OS starts.

Add the following registry entries to the Project.reg file, to launch the Remote Display application:

```
[HKEY_LOCAL_MACHINE\Startup]
                "Process2"="cerdisp -c"
                "Process2Delay"=dword:00002710       ; delay 10 seconds
```

With the registry entries added to launch the CoreCon component and Remote Display application, the code editor window on the VS2005 IDE should looks similar to Figure 19.



```
; Let Start/Settings/Control Panel Auto Expand
[HKEY_LOCAL_MACHINE\Explorer]
    "ExpandControlPanel"=dword:0


[HKEY_LOCAL_MACHINE\Windows CE Tools]
  "Platform"="{6383714b-899d-4563-acbf-260a52619a09}"

[HKEY_LOCAL_MACHINE\Startup]
    "Process1"="ConmanClient2.exe"
    "Process1Delay"=dword:00001388
    "Process2"="cerdisp -c"
    "Process2Delay"=dword:00002710
```

Fig. 19 - VS2005 IDE Code editor window – Project.reg

Other CE 6.0 Components

In addition to the components selected during the OS design wizard and the components automatically included in the OS design by the OS design wizard, additional components from the catalog can be included 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.

Following is a list of the VDX-6326 SBC I/O peripherals and the associated CE 6.0 device drivers and support components.

VDX-6326 SBC's Peripherals	**Windows Embedded CE 6.0 Drivers & Support Components**
IDE	ATAPI storage driver – This driver is provided by the Platform Builder, in the component catalog. The ATAPI driver component in the ICOP_VDX6326_60B BSP, when selected, enables the appropriate environment variables to include the ATAPI driver and support library, provided by Platform Builder, to the OS design project.
R6040 Ethernet	R6040 Ethernet driver – This is a third party driver provided as part of the ICOP_VDX6326_60B BSP by ICOP, to support the R6040 Ethernet controller, built-in with the Vortex86DX System-On-Chip.
Realtek-8100 Ethernet	Realtek-8139 Ethernet driver – This driver is provided by the Platform Builder, in the component catalog. The Realtek-8139 Ethernet driver component in the ICOP_VDX6326_60B BSP, when selected, enables the appropriate environment variables to include the Realtek RTL-8139 driver and support library, provided by Platform Builder, to the OS design project.
Serial Ports	Serial Port driver – This driver is provided by the Platform Builder, in the component catalog. The Serial Port driver components in the ICOP_VDX6326_60B BSP, when selected, enable the appropriate environment variables to include the Com16550 serial port driver, provided by the Platform Builder, to the OS design project, and configure the necessary registry entries to support each of the 4 serial ports.
SPI Flash Storage	SPI Flash driver – This is a third party driver provided as part of the ICOP_VDX6326_60B BSP by ICOP, to support the onboard 4MB SPI flash.
USB Audio	USB Audio driver – This is a third party driver provided as part of the ICOP_VDX6326_60B BSP by ICOP, to support the USB audio controller.
Z9s display	Z9s Display driver – This is a third party driver provided as part of the ICOP_VDX6326_60B BSP by ICOP, to support the Z9s display controller.
USB Ports (USB 2.0 Host)	USB host drivers (Driver available from Platform Builder's component catalog)
CompactFlash*[1]	ATAPI storage driver – This driver is provided by the Platform Builder, in the component catalog. The ATAPI driver component in the ICOP_VDX6326_60B BSP, when selected, enables the appropriate environment variables to include the ATAPI driver and support library, provided by Platform Builder, to the OS design project.

*[1] The CompactFlash (CF) slot on SBC is link to the IDE interface, and does NOT support hot-swap. When a CF card is plugged into the slot prior to power on, the system will detect and recognize the CF card as an IDE storage device. When booting to CE 6.0, the CF card will shows up as "Hard Disk".

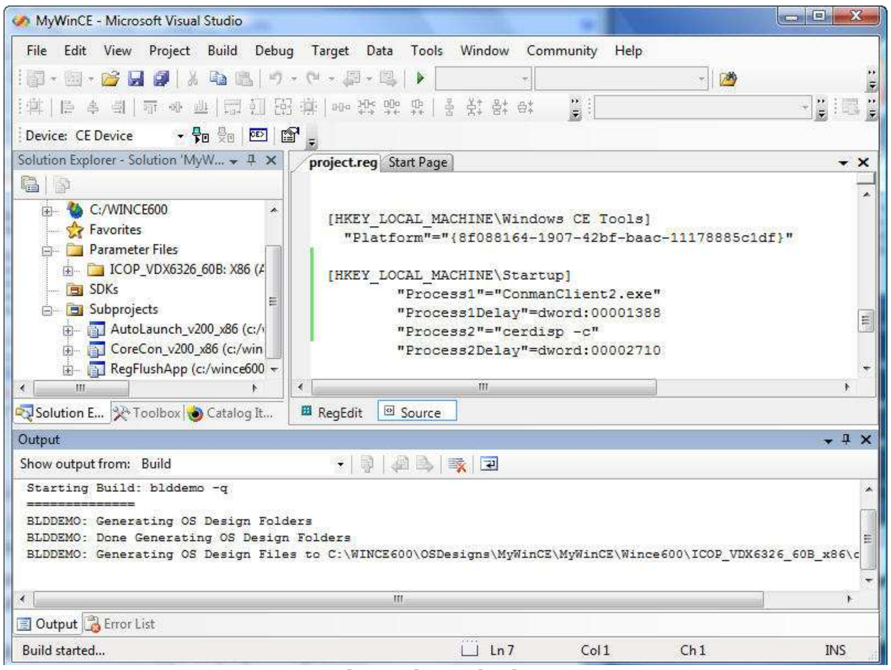
The SBC used to develop the exercise for this guide is configured to boot from an IDE flash attached to the SBC's 44-pin IDE interface. The IDE flash emulates as an IDE storage device and requires the ATAPI driver and FAT file system component to function.

To learn more about each of the components on the catalog, refer to the help document.

Generate CE 6.0 OS Runtime Image from the OS Design

After selecting all of the needed components for the OS design, we can continue and build the OS design to generate a CE 6.0 OS runtime image from the OS design.

From the VS2005 IDE, select **Build | Build Solution** to start the build process.



```
[HKEY_LOCAL_MACHINE\Windows CE Tools]
  "Platform"="{8f088164-1907-42bf-baac-11178885c1df}"

[HKEY_LOCAL_MACHINE\Startup]
        "Process1"="ConmanClient2.exe"
        "Process1Delay"=dword:00001388
        "Process2"="cerdisp -c"
        "Process2Delay"=dword:00002710
```

```
Starting Build: blddemo -q
===============
BLDDEMO: Generating OS Design Folders
BLDDEMO: Done Generating OS Design Folders
BLDDEMO: Generating OS Design Files to C:\WINCE600\OSDesigns\MyWinCE\MyWinCE\Wince600\ICOP_VDX6326_60B_x86\c
```

Fig. 20 - VS2005 IDE – OS design being built.

Depending on the speed and performance of the development workstation, the build process may take anywhere from 15 minutes to well over 30 minutes.

During the build process, the output tab on the VS2005 IDE displays compilation activities, as shown in Figure 20.

Important Note:

Don't use the "**Build and Sysgen"** and "**Rebuild and Clean Sysgen"** options. When one of these options is executed, it will delete some of the binary files that you don't have the source codes to rebuild. The only way to recover is to reinstall Platform Builder. While these build options are needed by some developer, they are dangerous to the developer who does not need them.

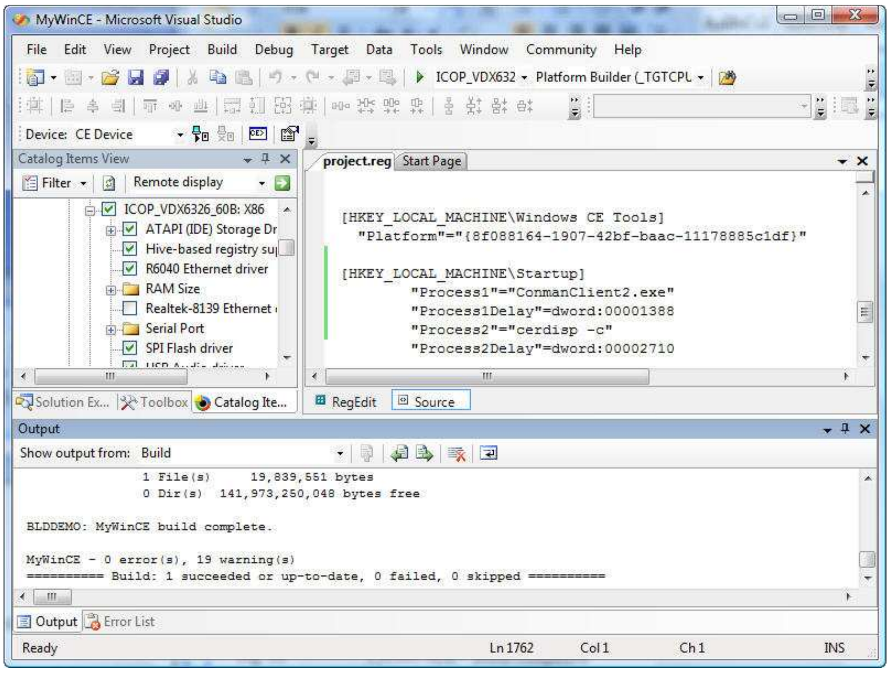
It's recommended to remove these two build options to avoid accidently clicking on them.

Work through the following steps to remove these two build options from the menu:

- From VS2005 IDE, select **Tools | Customize…** to bring up the Customize dialog.
- With the Customize dialog open, from the VS2005 IDE, click and hold the **Build | Advanced Build Commands | Build and Sysgen** option and drag it out to an empty space on the VS2005 IDE to remove it from the menu.
- Repeat the above step to remove **Build | Advanced Build Commands | Rebuild and Clean Sysgen** from the menu.

Build Complete – CE 6.0 OS Image Generated

When the build process is completed, the VS2005 IDE should look similar to the screen, as shown in Figure 21.



```
[HKEY_LOCAL_MACHINE\Windows CE Tools]
  "Platform"="{8f088164-1907-42bf-baac-11178885c1df}"

[HKEY_LOCAL_MACHINE\Startup]
        "Process1"="ConmanClient2.exe"
        "Process1Delay"=dword:00001388
        "Process2"="cerdisp -c"
        "Process2Delay"=dword:00002710
```

```
               1 File(s)     19,839,551 bytes
               0 Dir(s)  141,973,250,048 bytes free

BLDDEMO: MyWinCE build complete.

MyWinCE - 0 error(s), 19 warning(s)
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Fig. 21 - VS2005 IDE – Build completed

The Output tab on the VS2005 IDE shows the result from the build process.

When the build process ended with 1 or more error, the build process failed and will not generate an OS runtime image file.

When the build process ended with warnings and without error, the build process is completed with an OS runtime image file generated. The number of warning may vary depending on the selected components and installed QFE. In general, the warnings are non critical and do not impact system function and can be ignored.

As part of a good engineering practice, it's good to go through the warning messages to identify potential problem.

During the build process, files are copied and generated in the following build release directories:

- For OS design configured to generate an image in debug mode:
  `\WINCE600\OSDesigns\MyWinCE\MyWinCE\RelDir\ICOP_VDX6326_60B_x86_Debug`
- For OS design configured to generate an image in release mode:
  `\WINCE600\OSDesigns\MyWinCE\MyWinCE\RelDir\ICOP_VDX6326_60B_x86_Release`

Searching the above directories, with a successful build, there should be a **NK.BIN** file, which is the CE 6.0 OS runtime image file.

In the next section, we will cover connecting the VS2005 development workstation to the target device to download and launch the image built in this section onto the target device, using the VDX-6326 SBC as the target device.

Part 7 – Download OS Image to Target Device

Preparing the Development Workstation and VDX-6326 SBC

There are different methods to establish connectivity and download the CE 6.0 OS runtime image from the development workstation to the target device, through an Ethernet interface, serial port, USB or JTAG interface.

For the VDX-6326 SBC, Ethernet is used as the primary interface to establish connectivity to the development workstation. For the exercise in this guide, both the development workstation and the SBC are attached to the same Local Area Network with DHCP service to provide IP address dynamically.

It's possible to establish connectivity using a Local Area Network without DHCP service, with static IP addresses.

Please refer to Appendix A and B for more information about connectivity options between the SBC and development workstation.

If you experience problem establishing connection, disable the firewall on the development workstation. The firewall may be blocking the connection.

The SBC that comes with the Vortex86DX-SPARK hardware kit includes an IDE bootable flash storage, configured to boot to DOS using FAT file system. The following software components are preconfigured on the IDE flash storage:

- Autoexec.bat
  This is the startup batch file for the DOS operating system, and is executed each time the operating system is launched.

- Config.sys
  This is the startup configuration file for the DOS operating system.

- Eboot.bin
  This is the Ethernet boot loader, needed to establish connectivity between the development workstation and the target device to download CE 6.0 OS runtime image from the development workstation. The Eboot.bin Ethernet boot loader needs to be launched by another boot loader.

- Loadcepc.exe
  This is the DOS boot loader, used for development purpose. The Loadcepc boot loader can launch CE 6.0 OS runtime image stored on the target device's local storage, NK.bin. It's also used to launch the Eboot.bin Ethernet boot loader to establish connectivity with the development workstation and broadcast bootme request to download the CE 6.0 OS runtime image from the development workstation.

- NK.bin
  This is a CE 6.0 OS runtime image.

After power up, the VDX-6326 SBC boot to DOS and launches a selection menu with the following options:

1. Load OS image with from local storage
2. Load OS image from development station with DHCP service
3. Load OS image from development station with Static IP 192.168.2.232
4. Clean Boot (no commands)

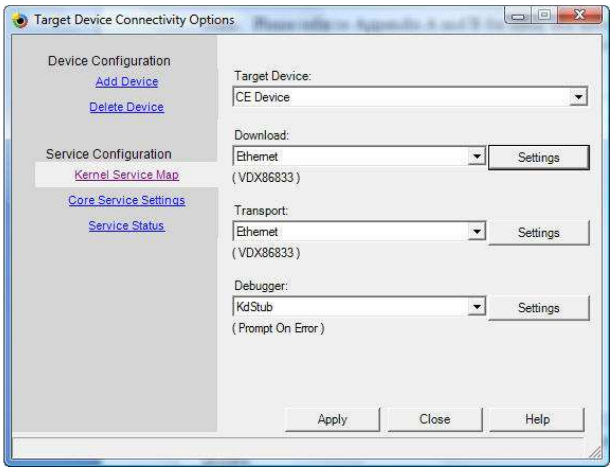
We will use option 2, Load OS image from development station with DHCP service, for the exercise in this guide.

> Development Station with static IP address:
>
> If you are working in an environment without DHCP service, using option 3 and configure your development machine with a proper static IP address. The following is the recommended static IP configuration for the development workstation:
>
> | | |
> |---|---|
> | IP Address: | 192.168.2.132 |
> | Subnet mask: | 255.255.255.0 |

Configure Target Device Connectivity Options

The image generated from the previous session is ready to be downloaded to the target device. Before the download can take place, we need to establish connectivity between the target device and the development workstation to accomplish the task.

From VS2005 IDE, select **Target | Connectivity Options…** to bring up the **Target Device Connectivity Options** screen, as shown in Figure 22.



Target Device Connectivity Options

Device Configuration
Add Device
Delete Device

Service Configuration
Kernel Service Map
Core Service Settings
Service Status

Target Device: CE Device

Download: Ethernet (VDX86833) Settings

Transport: Ethernet (VDX86833) Settings

Debugger: KdStub (Prompt On Error) Settings

Apply Close Help

Fig. 22 - Target Device Connectivity Options

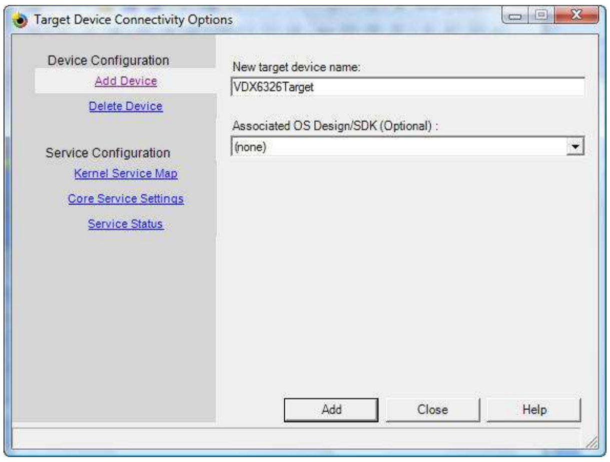
A connection can be establish by modifying the default **CE Device** settings.

However, when working with multiple devices, it’s helpful to create unique device profile for the target device. The unique device profile established for each target device can help save time during the development process.

Work through the following steps to create a device connectivity profile for the VDX-6326 SBC, and use VDX6326Target as the name for this profile.

Add New Target Device Profile

From the **Target Device Connectivity Options** screen, click on Add Device to bring up the screen to add a new target device profile as shown in Figure 23.



New target device name: VDX6326Target

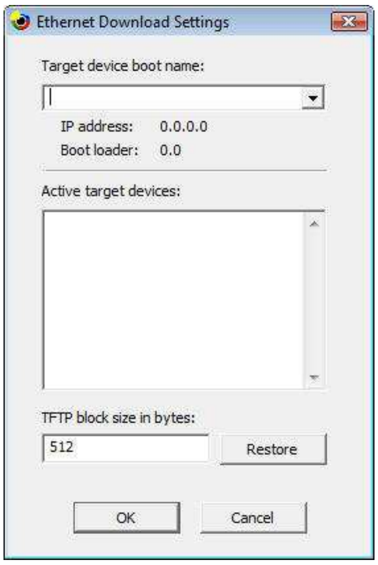
Associated OS Design/SDK (Optional) : (none)

Fig. 23 - Target Device Connectivity Options – Add new device

- Enter **VDX6326Target** as the new target device
- Click on the **Add** button to continue
- Select **Ethernet** for the **Download** and **Transport** options
- Select **KdStub** for the **Debugger** option

Establish Connectivity with the SBC

To establish connectivity between the development workstation and SBC using the VDX6326Target device profile, from the **Target Device Connectivity Options** screen, clicks on the top most **Settings** buttons, to bring up the **Ethernet Download Settings** screen, as shown in Figure 24.



Target device boot name:

IP address: 0.0.0.0

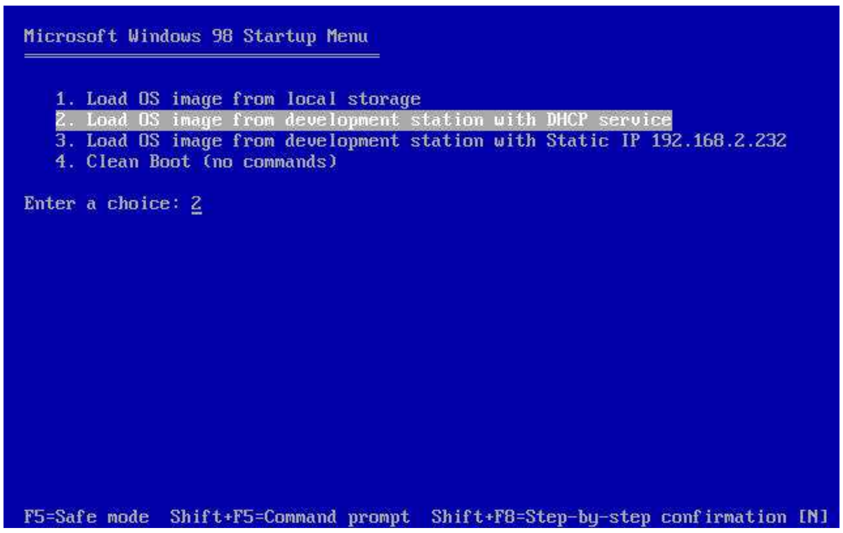
Boot loader: 0.0

Active target devices:

TFTP block size in bytes: 512

Fig. 24 - Ethernet Download Settings

Apply power to the VDX-6326 SBC. The SBC is preconfigured to boot to DOS and launch a selection menu with multiple options, as shown in Figure 25.



```
Microsoft Windows 98 Startup Menu
═════════════════════════════════

   1. Load OS image from local storage
   2. Load OS image from development station with DHCP service
   3. Load OS image from development station with Static IP 192.168.2.232
   4. Clean Boot (no commands)

Enter a choice: 2




F5=Safe mode  Shift+F5=Command prompt  Shift+F8=Step-by-step confirmation [N]
```

Fig. 25 - DOS selection menu from the SBC

Option 1:	Load OS image from local storage.
Option 2:	Load OS image from development station with DHCP service
Option 3:	Load OS image from development station with Static IP 192.168.2.232
Option 4:	Clean Boot (no commands)

When option 1 is selected, the system launches the NK.bin OS image file from local storage.

When option 2 is selected, the system launches Eboot.bin, request IP address available DHCP server and sent request to download OS image from Platform Builder development station.

When option 3 is selected, the system launches Eboot.bin with static IP address (192.168.2.232) and sent request to download OS image from Platform Builder development station.

When option 4 is selected, the system boot to the DOS command line.

For the setup where the development workstation and SBC are connected to the same LAN with DHCP service, select option 2 to download the runtime image from the development workstation.

```
2.  Load OS image from development station with DHCP service.
```

For the setup where the development workstation and SBC are connected to the same LAN without DHCP service, select option 3.

```
3.  Load OS image from development station with Static IP 192.168.2.232
```

When connecting the SBC to the development station using a cross-over Ethernet cable, configure the development station's IP address to 192.168.2.132, with subnet mask set to 255.255.255.0, and use option 3 to send request for downloading the OS image.

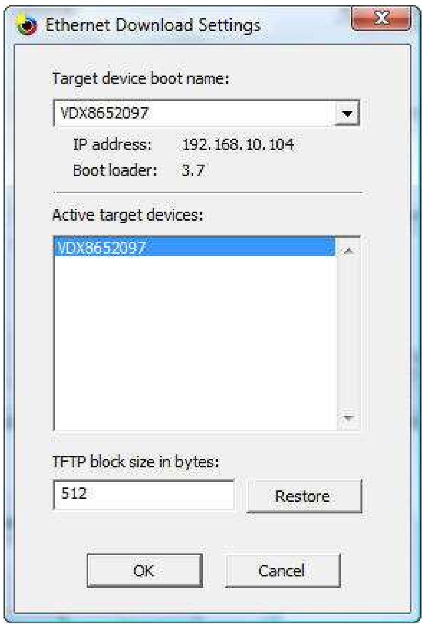
The development workstation and SBC used to develop the exercises for this jump start guide are both connected to the same local LAN with DHCP service, and use option 2 to download runtime image from the development workstation.

When option 2 is selected, the SBC launches the Loadcepc.exe executable, a DOS boot loader with the following command line parameters to launch the Ethernet boot loader, eboot.bin:

```
loadcepc /C:1 /e:%NET_IOBASE%:%NET_IRQ%:%NET_IP% eboot.bin
```

The eboot.bin is an Ethernet boot loader. When launched by the Loadcepc.exe boot loader, it sends a request to the DHCP server to acquire an IP address and sends a bootme message to the Platform Builder development workstation to initiate the download process.

After receiving the bootme message from the SBC, the SBC's device ID will shows up on the Active target devices list on the **Ethernet Download Settings** screen, as shown in Figure 26.


Ethernet Download Settings

Target device boot name: VDX8652097

IP address: 192.168.10.104

Boot loader: 3.7

Active target devices: VDX8652097

TFTP block size in bytes: 512

Fig. 26 - Ethernet Download Settings

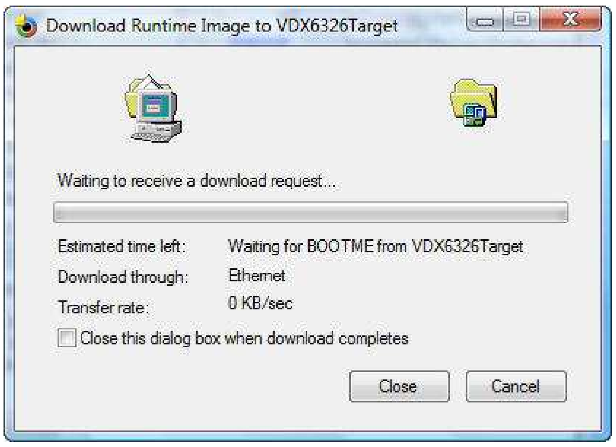
- Click and highlight the device ID listed in the **Active Devices** window.
- Click **Ok** to continue.

> In the environment where there are multiple target device connected to the same network segment booting up at the same time, there may be multiple device IDs listed in the Active Devices windows. To identify the SBC you are working with, make sure it's the only one booting and sending the bootme request.

On the **Target Device Connectivity Options** screen, click on **Apply** button follow by the **Close** button to save the settings and close the **Target Device Connectivity Options** screen.

Downloading OS Runtime Image to the VDX-6326 SBC

You are now ready to download the CE 6.0 OS runtime image, generated from the OS design during the earlier steps, to the SBC. From VS2005 IDE select **Target | Attach Device** to initiate the download process and bring up the "**Download Runtime Image to VDX6326Target**" screen, as shown in Figure 27.



Download Runtime Image to VDX6326Target

Waiting to receive a download request...

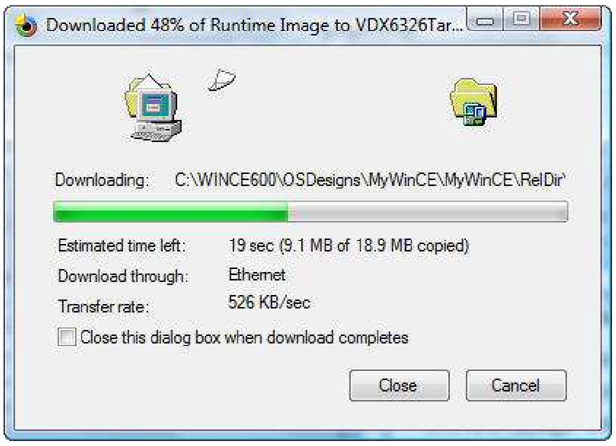
Estimated time left: Waiting for BOOTME from VDX6326Target
Download through: Ethernet
Transfer rate: 0 KB/sec

Fig. 27 - Download Runtime Image / waiting for bootme request

At this point, if the SBC is still sending the bootme message, the download will start briefly. Otherwise, you need to reset the SBC and select the same option from the DOS selection menu again.

After the eboot.bin Ethernet boot loader is launched by the Loadcepc.exe boot loader, it broadcasts the bootme messages to the CE 6.0 Platform Builder development IDE repeatedly, and stop the broadcast after about two minutes.

After SBC sends the bootme request and established connectivity with the development workstation, you will see activities on the **Download Runtime Image** screen, showing the CE 6.0 OS image being downloaded to the SBC, as shown in Figure 28.



Downloaded 48% of Runtime Image to VDX6326Tar...

Downloading: C:\WINCE600\OSDesigns\MyWinCE\MyWinCE\RelDir\

Estimated time left: 19 sec (9.1 MB of 18.9 MB copied)
Download through: Ethernet
Transfer rate: 526 KB/sec

Fig. 28 - Download Runtime Image / downloading

After the image download process is completed, the SBC will launch the CE 6.0 runtime image it just received. Be patient… It takes a few moments for the image to come up.

When the Windows CE screen is displayed on video monitor, it's an indication the booting process is completed, as shown in Figure 29.

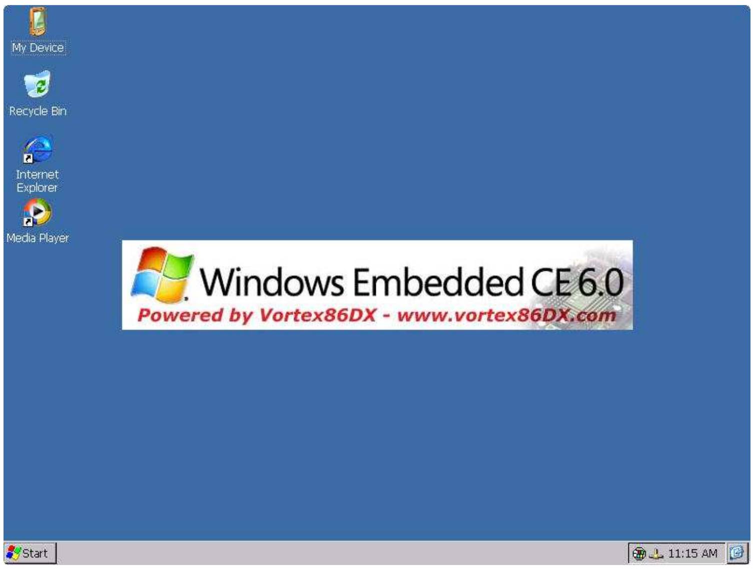


Fig. 29 - Windows Embedded CE 6.0 desktop running on the VDX-6326 SBC

Deploy the CE 6.0 OS Image to the VDX-6326 SBC

In the previous step, when the CE 6.0 image is downloaded and launched on the SBC, the image is placed directly to the SBC's system memory (RAM) and is not saved to the local storage. After the SBC gone through power reset, the image will not remain. To deploy a CE 6.0 OS image onto the SBC's local storage, you need configure the local storage with an appropriate bootloader and place a copy the OS image, the **NK.bin** file, to SBC's local flash storage.

The **NK.bin** runtime image file is generated from the OS design in the following directory:

```
C:\WINCE600\OSDesigns\MyWinCE\MyWinCE\RelDir\ICOP_VDX6326_60B_x86_Release
```

There are multiple methods to copy the CE 6.0 image to the SBC's local storage.

1. Use a USB bootable flash storage.

   The VDX-6326 SBC can be configured to boot from an USB bootable storage device. Refer to Appendix E for setup information.

   Using a USB bootable flash storage, copy the CE 6.0 OS image file, **NK.bin**, from *\WinCE600\OSDesigns\MyWinCE\MyWinCE\RelDir\ICOP_VDX6326_60B_x86_Release* directory to the USB bootable flash storage. Boot the SBC with the USB bootable flash storage and copy the NK.bin file to the SBC IDE flash storage's root directory (over written the existing **NK.bin** file).

2. Place the **NK.bin** image file, from the development workstation, to a shared network file directory. With the CE 6.0 OS image downloaded to the SBC from the development workstation running, copy the **NK.bin** image from a shared network file directory to the root of the SBC's file system or "Hard Disk" folder, over writing the existing Nk.bin file.

The VDX-6326 SBC comes with the jump start kit is configured with DOS and the Loadcepc.exe boot loader for development purpose.

To deploy CE 6.0 OS runtime image to the target device for distribution, you need to configure the target device's storage with the appropriate file system and boot loader to launch the runtime image and copy the CE 6.0 OS runtime image to the file system.

When deploying a CE 6.0 device for distribution, a dedicated boot loader is needed to launch the CE OS runtime image from the target device's local storage.

The BIOSLoader is provided as part of the public code, including the source code, which you can modify and use as the boot loader for the final product. The BIOSLoader is provided in the following folder:

- \WINCE600\Platform\CEPC\SRC\Bootloader\BIOSLoader

The DOS operating system and Loadcepc boot loader, provided as part of this jump start kit, are used for development purpose only and not for distribution.

Unless you are planning to configure the SBC to function in distribution mode, do not configure the SBC's internal IDE flash storage with BIOSLoader. Doing so will wipe out the existing DOS operating system and preconfigured files.

Once the DOS operating system and preconfigured files are wiped out or damaged, you need to boot SBC to DOS using an USB floppy or USB flash storage and recover the image manually, by formatting the SBC's internal IDE flash to boot to DOS and copy the original files from the jump start kit CD-ROM.

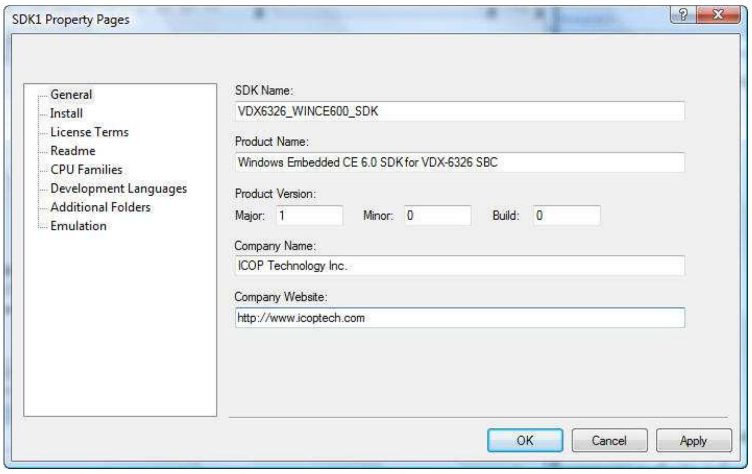
Refer to Appendix-I to recover files from the jump start kit CD-ROM.

Part 8 – Windows Embedded CE 6.0 SDK

In the previous steps, we worked through the steps to create, customize a CE 6.0 OS design, generated OS runtime image from the OS design and downloaded the runtime image to the target device. In this section, we will work through the steps to create and configure a SDK for the OS design, which is needed to develop application using Visual Studio 2005 or 2008.

Create and Configure Windows Embedded CE 6.0 SDK

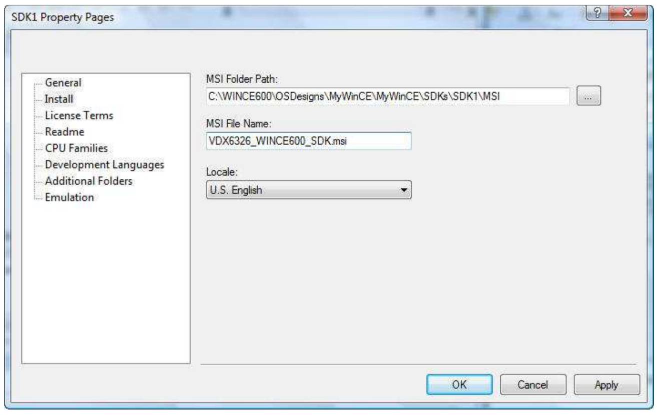
From VS2005 IDE, select **Project | Add New SDK…** to bring up the **SDK Property Pages** screen, as shown in Figure 30.



SDK Name: VDX6326_WINCE600_SDK
Product Name: Windows Embedded CE 6.0 SDK for VDX-6326 SBC
Product Version: Major: 1 Minor: 0 Build: 0
Company Name: ICOP Technology Inc.
Company Website: http://www.icoptech.com

Fig. 30 - SDK Property Page

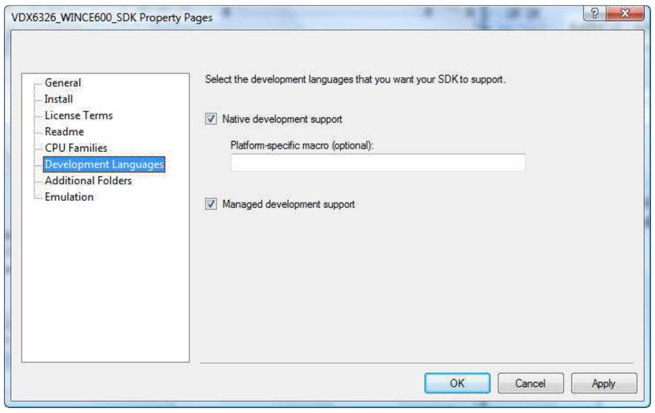
- Enter **VDX6326_WINCE600_SDK** as the name for the SDK
- Fill in the company name and company website information
- On the left side of **SDK Property Page**, click on **Install** and enter path and file name for the MSI file on the right side. Use the default path, and enter **VDX6326_WINCE600_SDK.msi** as the file name, as shown in Figure 31.



MSI Folder Path: C:\WINCE600\OSDesigns\MyWinCE\MyWinCE\SDKs\SDK1\MSI
MSI File Name: VDX6326_WINCE600_SDK.msi
Locale: U.S. English

Fig. 31 - SDK Property Page

- On the left side of **SDK Property Page**, click on Development Languages and select both the **Native development support** and **Managed development support** check box on the right, as shown in Figure 32**.**


VDX6326_WINCE600_SDK Property Pages

- General
- Install
- License Terms
- Readme
- CPU Families
- Development Languages
- Additional Folders
- Emulation

Select the development languages that you want your SDK to support.

[✓] Native development support

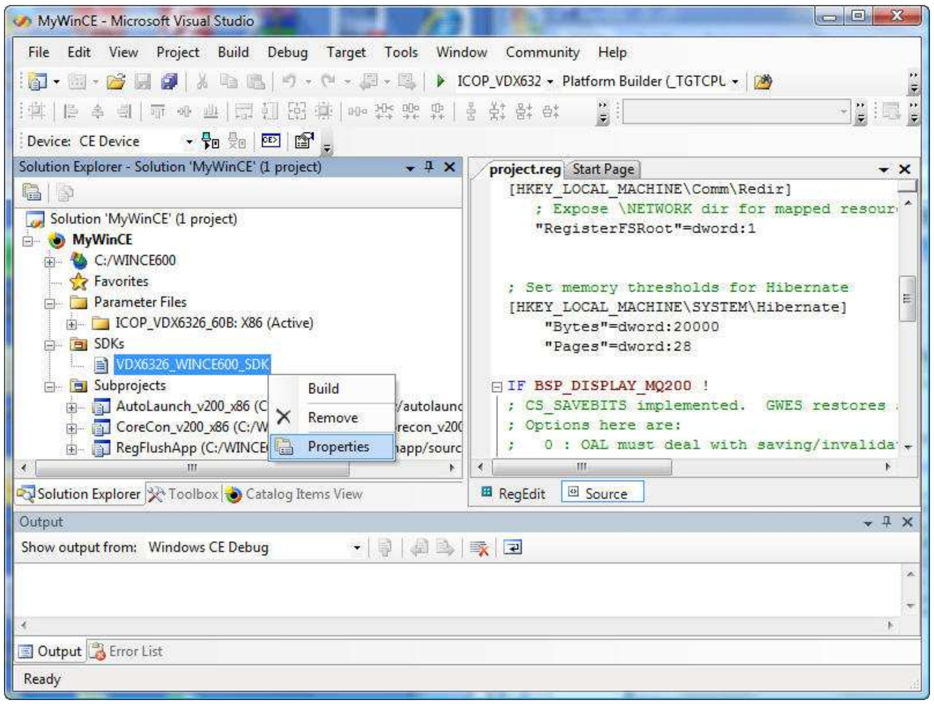
Platform-specific macro (optional):

[✓] Managed development support

Fig. 32 - SDK Property Page

- Click on the **Apply** button and then the **OK** button to complete the Add New SDK process.

To make changes to the SDK before building, from VS2005 IDE, select the **Solution** tab, expand the **SDKs** folder, right click on **VDX6326_WINCE600_SDK** and select **Properties** to bring up the **SDK Property Pages**, as shown in Figure 33.


MyWinCE - Microsoft Visual Studio

Solution Explorer - Solution 'MyWinCE' (1 project)

```
Solution 'MyWinCE' (1 project)
  MyWinCE
    C:/WINCE600
    Favorites
    Parameter Files
      ICOP_VDX6326_60B: X86 (Active)
    SDKs
      VDX6326_WINCE600_SDK
    Subprojects
      AutoLaunch_v200_x86 (C
      CoreCon_v200_x86 (C:/W
      RegFlushApp (C:/WINCE
```

project.reg

```
[HKEY_LOCAL_MACHINE\Comm\Redir]
   ; Expose \NETWORK dir for mapped resour
   "RegisterFSRoot"=dword:1


; Set memory thresholds for Hibernate
[HKEY_LOCAL_MACHINE\SYSTEM\Hibernate]
    "Bytes"=dword:20000
    "Pages"=dword:28

IF BSP_DISPLAY_MQ200 !
; CS_SAVEBITS implemented.  GWES restores
; Options here are:
;    0 : OAL must deal with saving/invalida
```

Fig. 33 - VS2005 IDE / Edit SDK

Build Windows Embedded CE 6.0 SDK

To build and generate the SDK installation file, select **Build | Build All SDKs…** from the VS2005 IDE.

It's possible to build and generate the SDK installation file with the following steps.

- From VS2005 IDE, select the **Solution** tab, expand the **SDKs** folder, right click on **VDX6326_WINCE600_SDK** and select **Build**

After the build process is completed, the SDK with the file, **VDX6326_WINCE600_SDK.msi**, is generated in the following folder.

```
 \WINCE600\OSDesigns\MyWinCE\MyWinCE\SDKs\SDK1\MSI\
```

Install this SDK to the Visual Studio development workstation to development CE 6.0 application for the VDX-6326 SBC.

Part 9 – Managed Code Application with Visual Studio 2005

VS2005 and VS2008 can be used to develop native and managed code applications for CE 6.0.

Notes: Steps to develop Managed code application using VS2008 are similar to VS2005.

In this section, we will cover the following:

- Develop a C# managed code application using VS2005.
- Establish connectivity between the development workstation and SBC using CoreCon.
- Download the application to the SBC for testing and debug.

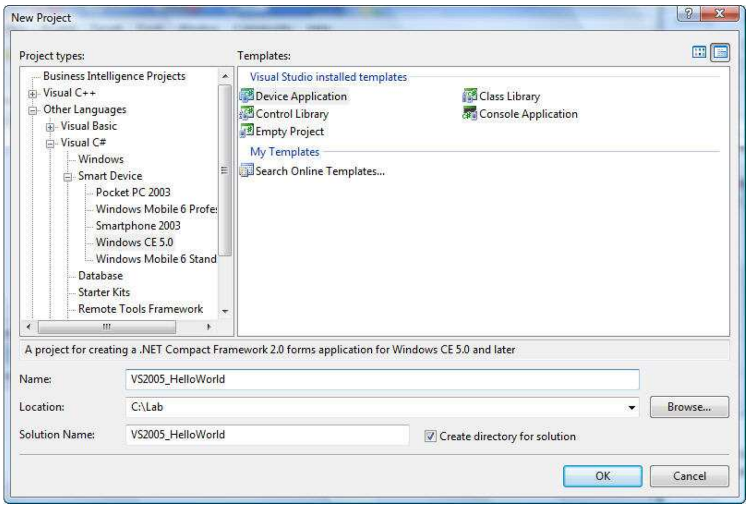
You can use similar steps to develop application with VS2008.

A managed code application with Visual Studio 2008 exercise is provided on Appendix M, toward the end of this guide.

Project files for the VS2005 and VS2008 managed code exercises are provided in the \Application Sample\ folder on the jump start CD.

Step 1: Create a New Visual Studio 2005 C# Project

Launch VS2005, select **File | New | Project** from the VS2005 IDE to launch the New Project wizard, as shown in Figure 34.



A project for creating a .NET Compact Framework 2.0 forms application for Windows CE 5.0 and later

Name: VS2005_HelloWorld

Location: C:\Lab

Solution Name: VS2005_HelloWorld

[✓] Create directory for solution

Fig. 34 - VS2005 IDE / New C# managed code Project

On the left hand side, expand the "\Other Languages\Visual C#\Smart Device\" folder to select "Windows CE 5.0" project type. On the right side of the screen, select "Device Application", enter "VS2005_HelloWorld" as the project name and click OK to continue.

The new project wizard will generate the initial project files, which include a blank form. Let's add some simple code to the application.

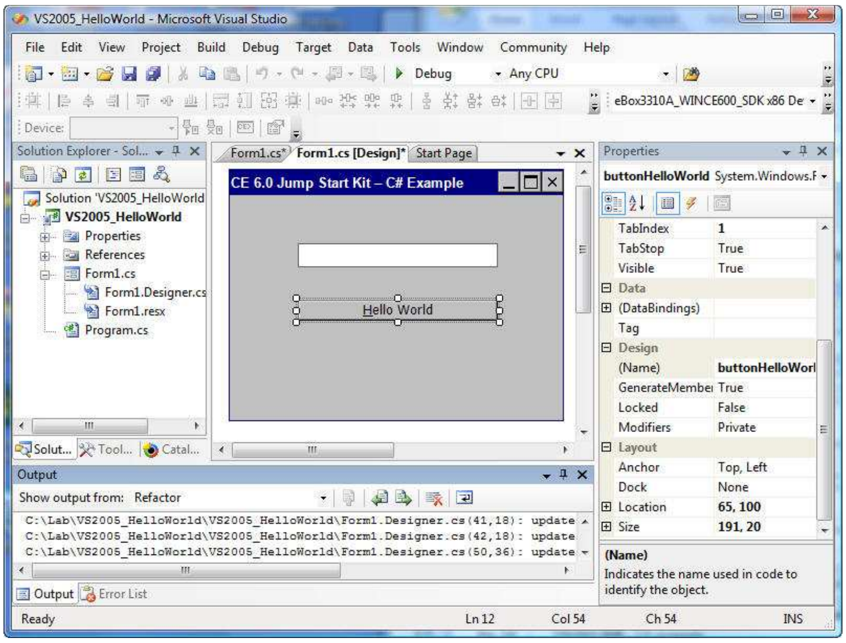
- Resize the Form1 to a smaller size (320x240) to make it easy to see the application when deploy on the CE 6.0 target device.

- Remove the mainMenu1 component placed onto Form1 by the wizard.
- Change the Form caption to “CE 6.0 Jump Start Kit – C# Example”.
- Add a text-box to Form1 and change the name to textHelloWorld, clear the content in the text-box and place the text-box to the center of Form1.
- Add a button to Form1, change the name to buttonHelloWorld and change the text on the button’s caption to “Hello World” and place the button to the center of Form1, just below the textHelloWorld text-box

  Add the following code to the “buttonHelloWorld_Click” event.

```
textHelloWorld.Text = "Hello World!";
textHelloWorld.Text = "2nd Hello World!";
textHelloWorld.Text = "3rd Hello World!";
textHelloWorld.Text = "Last Hello World!";
```

VS2005 IDE screen should look similar to the following screen shot, as shown in Figure 35.


VS2005_HelloWorld - Microsoft Visual Studio
File Edit View Project Build Debug Target Data Tools Window Community Help
Debug
Any CPU
eBox3310A_WINCE600_SDK x86 De
Device:
Solution Explorer - Sol...
Solution 'VS2005_HelloWorld
VS2005_HelloWorld
Properties
References
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
Form1.cs* Form1.cs [Design]* Start Page
CE 6.0 Jump Start Kit – C# Example
Hello World
Solut... Tool... Catal...
Output
Show output from: Refactor
Output Error List
Ready Ln 12 Col 54 Ch 54 INS

```
C:\Lab\VS2005_HelloWorld\VS2005_HelloWorld\Form1.Designer.cs(41,18): update
C:\Lab\VS2005_HelloWorld\VS2005_HelloWorld\Form1.Designer.cs(42,18): update
C:\Lab\VS2005_HelloWorld\VS2005_HelloWorld\Form1.Designer.cs(50,36): update
```

Properties

buttonHelloWorld System.Windows.F

TabIndex	1
TabStop	True
Visible	True
Data	
(DataBindings)	
Tag	
Design	
(Name)	buttonHelloWorl
GenerateMembe	True
Locked	False
Modifiers	Private
Layout	
Anchor	Top, Left
Dock	None
Location	65, 100
Size	191, 20

(Name)
Indicates the name used in code to identify the object.

Fig. 35 - VS2005 IDE / C# example

From the VS2005 IDE, select **Build | Build Solution** to compile and generate executable binary for the project.

Step 2: Preparing SBC to Connect to Visual Studio 2005 IDE

To perform this portion of the exercise, CE 6.0 image configured and built during the previous sections of this guide must be downloaded and running on the SBC, and the VDX6326_WINCE600_SDK must be installed to the development workstation.

CoreCon is used to establish connectivity between the SBC and development workstation’s VS2005 IDE.

To establish CoreCon connectivity, the following 5 CoreCon files need to be copied to the "\Windows" folder on SBC:

- Clientshutdown.exe
- ConmanClient2.exe
- CMaccept.exe
- eDbgTL.dll
- TcpConnectionA.dll

These files are installed to the following directory on the VS2005 development workstation, as part of the VS2005 installation:

```
\Program Files\Common Files\Microsoft Shared\CoreCon\1.0\Target\wce400\
```

CoreCon components supporting different type of processors are provided. There are multiple sub folders under the above directory, with name corresponding to supported processor. CoreCon files supporting the VDX-6326 SBC, built with x86 CPU, are in the \x86 sub folder.

> There are different versions of CoreCon files, installed to the development workstation as part of the Visual Studio 2005 and Visual Studio 2008 installation.
>
> CE 6.0 runtime image built with a different version of CoreCon files from the version on the development workstation will not be able to establish connectivity with the development workstation.

In the earlier section, during the customizing OS design steps, the CoreCon_v200_x86 component is added to the OS design. By adding this component, the same version of CoreCon files, installed to the development workstation as part of the Visual Studio installation, are included in the OS design and compiled as part of the CE 6.0 OS runtime image generated from the OS design.

To establish connectivity between the development workstation's VS2005 IDE and the SBC, we need to know the SBC's IP address. Work through the following steps to find the IP address for the SBC:

- With CE 6.0 running on the SBC, click on **Start | Run** from CE 6.0 desktop with the **CMD** command to open a console command window, as shown in Figure 36.

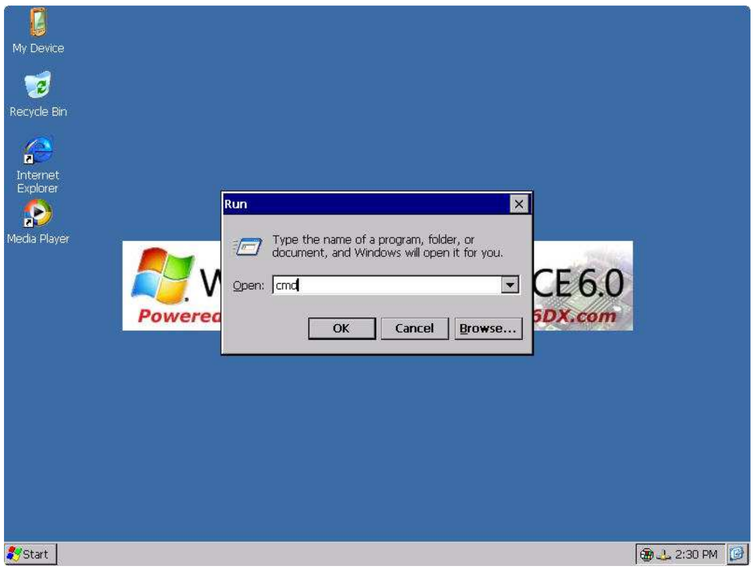
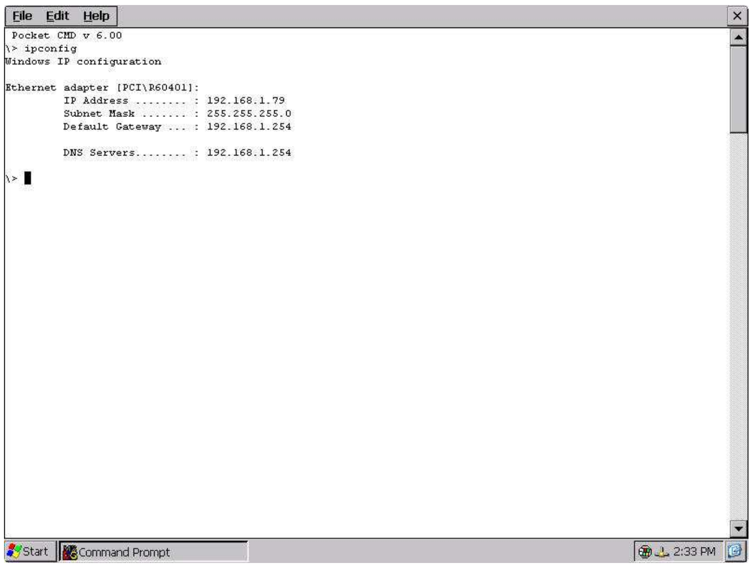


Fig. 36 - CE 6.0 desktop – Launching the CMD command

- From within the console command window, type **IpConfig** to show the IP address for the SBC, as shown in Figure 37.

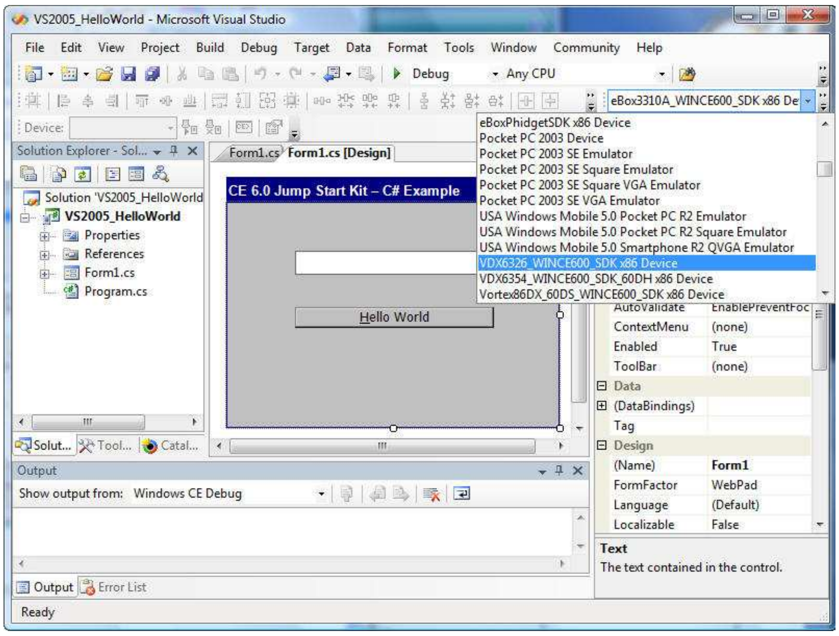


```
 Pocket CMD v 6.00
\> ipconfig
Windows IP configuration

Ethernet adapter [PCI\R60401]:
        IP Address ........ : 192.168.1.79
        Subnet Mask ....... : 255.255.255.0
        Default Gateway ... : 192.168.1.254

        DNS Servers........ : 192.168.1.254

\> 
```

Fig. 37 - CE 6.0 console command window showing the IP address information

Now that we have the IP address for the SBC, let's move to VS2005 IDE to configure device settings. In order for this to work, both the VS2005 development workstation and SBC must be connected to the same LAN segment and acquire their IP address from the same DHCP server.

From the VS2005 IDE, set the target device to "**VDX6326_WINCE600_SDK x86 Device**", as shown in Figure 38.


VS2005_HelloWorld - Microsoft Visual Studio
File Edit View Project Build Debug Target Data Format Tools Window Community Help
Debug
Any CPU
Device:
Solution Explorer - Sol...
Form1.cs
Form1.cs [Design]
CE 6.0 Jump Start Kit – C# Example
Hello World
Solut...
Tool...
Catal...
Output
Output
Error List
Ready

eBox3310A_WINCE600_SDK x86 De

eBoxPhidgetSDK x86 Device
Pocket PC 2003 Device
Pocket PC 2003 SE Emulator
Pocket PC 2003 SE Square Emulator
Pocket PC 2003 SE Square VGA Emulator
Pocket PC 2003 SE VGA Emulator
USA Windows Mobile 5.0 Pocket PC R2 Emulator
USA Windows Mobile 5.0 Pocket PC R2 Square Emulator
USA Windows Mobile 5.0 Smartphone R2 QVGA Emulator
VDX6326_WINCE600_SDK x86 Device
VDX6354_WINCE600_SDK_60DH x86 Device
Vortex86DX_60DS_WINCE600_SDK x86 Device

Solution 'VS2005_HelloWorld
VS2005_HelloWorld
Properties
References
Form1.cs
Program.cs

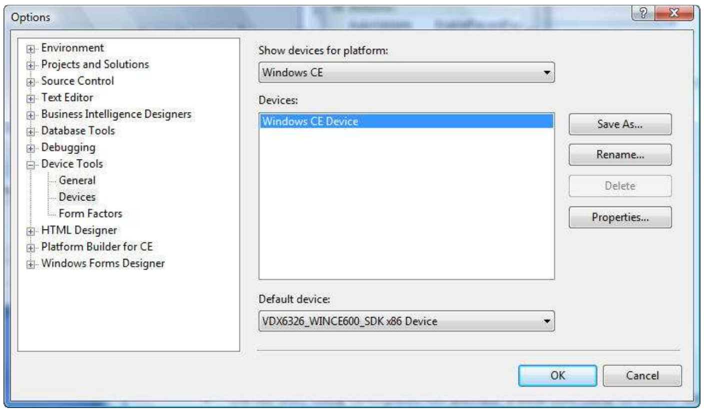
Show output from: Windows CE Debug

AutoValidate	EnablePreventFoc
ContextMenu	(none)
Enabled	True
ToolBar	(none)
Data	
(DataBindings)	
Tag	
Design	
(Name)	Form1
FormFactor	WebPad
Language	(Default)
Localizable	False

Text
The text contained in the control.

Fig. 38 - VS2005 IDE / select target device

From VS2005 IDE, select **Tools | Options…** to bring up the following configuration screen, as shown in Figure 39.



Environment
Projects and Solutions
Source Control
Text Editor
Business Intelligence Designers
Database Tools
Debugging
Device Tools
General
Devices
Form Factors
HTML Designer
Platform Builder for CE
Windows Forms Designer

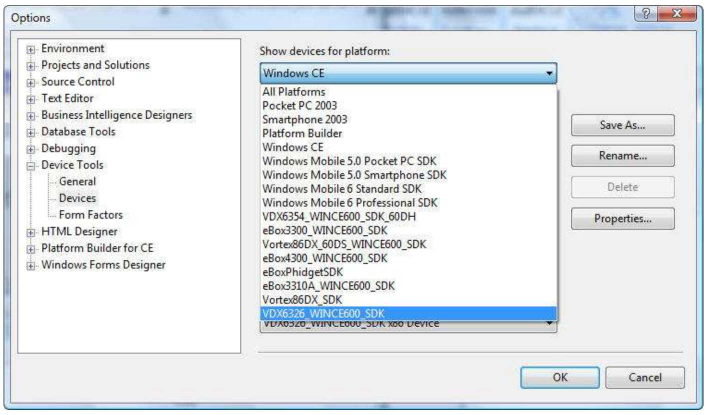
Show devices for platform: Windows CE

Devices: Windows CE Device

Default device: VDX6326_WINCE600_SDK x86 Device

Fig. 39 - VS2005 Tools Options

- On the left, click to expand the **Device Tools** node and select **Devices**.
- On the right, select **VDX6326_WINCE600_SDK** from the list of available platform in the **Show devices for platform** combo text box, as shown in Figure 40.


Options

Environment
Projects and Solutions
Source Control
Text Editor
Business Intelligence Designers
Database Tools
Debugging
Device Tools
General
Devices
Form Factors
HTML Designer
Platform Builder for CE
Windows Forms Designer

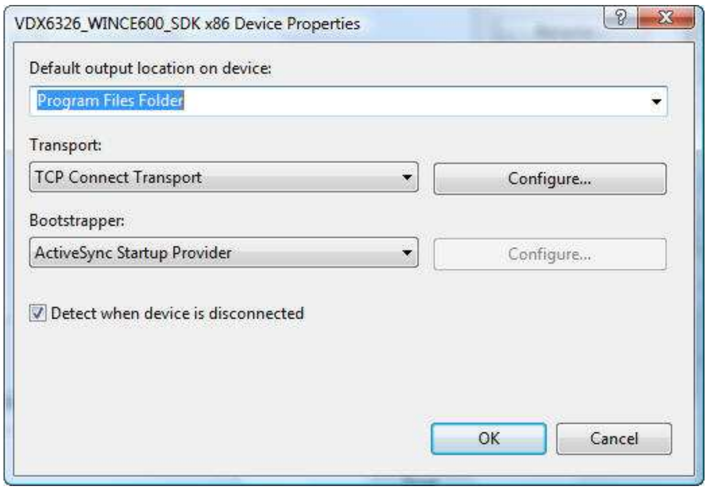
Show devices for platform:
Windows CE

All Platforms
Pocket PC 2003
Smartphone 2003
Platform Builder
Windows CE
Windows Mobile 5.0 Pocket PC SDK
Windows Mobile 5.0 Smartphone SDK
Windows Mobile 6 Standard SDK
Windows Mobile 6 Professional SDK
VDX6354_WINCE600_SDK_60DH
eBox3300_WINCE600_SDK
Vortex86DX_60DS_WINCE600_SDK
eBox4300_WINCE600_SDK
eBoxPhidgetSDK
eBox3310A_WINCE600_SDK
Vortex86DX_SDK
VDX6326_WINCE600_SDK

Save As...
Rename...
Delete
Properties...
OK
Cancel

Fig. 40 - VS2005 Tools Options

- Click on the **Properties** button to bring up **VDX6326_WINCE600 x86 Device Properties** setting screen, as shown in Figure 41.


VDX6326_WINCE600_SDK x86 Device Properties

Default output location on device:
Program Files Folder

Transport:
TCP Connect Transport
Configure...

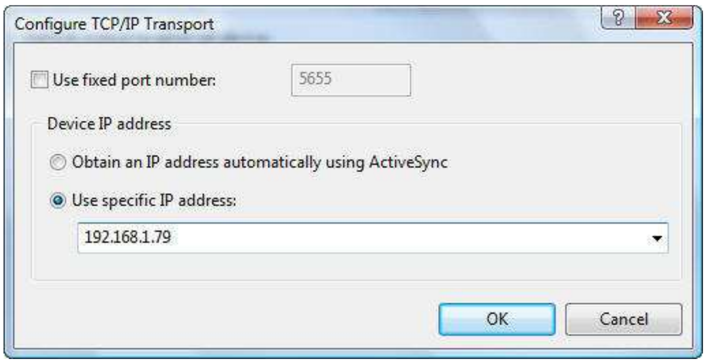
Bootstrapper:
ActiveSync Startup Provider
Configure...

☑ Detect when device is disconnected

OK
Cancel

Fig. 41 - VDX6326_WINCE600 x86 Device properties

- Click on the **Configure** button to bring up **Configure TCP/IP Transport** screen.
- Select **Use specific IP address** and enter IP address for the SBC, as shown in Figure 42.


Configure TCP/IP Transport

☐ Use fixed port number: 5655

Device IP address
○ Obtain an IP address automatically using ActiveSync
● Use specific IP address:
192.168.1.79

OK
Cancel

Fig. 42 - Configure TCP/IP Transport / Set device IP Address

- Click the **OK** button to save device IP address setting.
- Click the **OK** button on the **VDX6326_WINCE600_SDK x86 Device Properties** screen to close the screen.
- Click the **OK** button on the **Options** screen to close the screen.

Step 3: Connecting SBC to Visual Studio 2005 IDE using CoreCon

To initiate connectivity between the development workstation's VS2005 IDE and SBC using CoreCon, we need to launch the **ConmanClient2.exe** and **cMaccept.exe** CoreCon components from the SBC, with CE 6.0 running. This is a cumbersome process and has to perform at least once for each time the SBC reset power.

To make it easier to establish CoreCon connectivity, CoreCon files were added to the OS design and included to the OS runtime image. During customizing the OS design step, we added the CoreCon_v200_x86 component, AutoLaunch_v200_x86 component and entered registry entries to launch the CoreCon component automatically when the CE 6.0 OS starts.

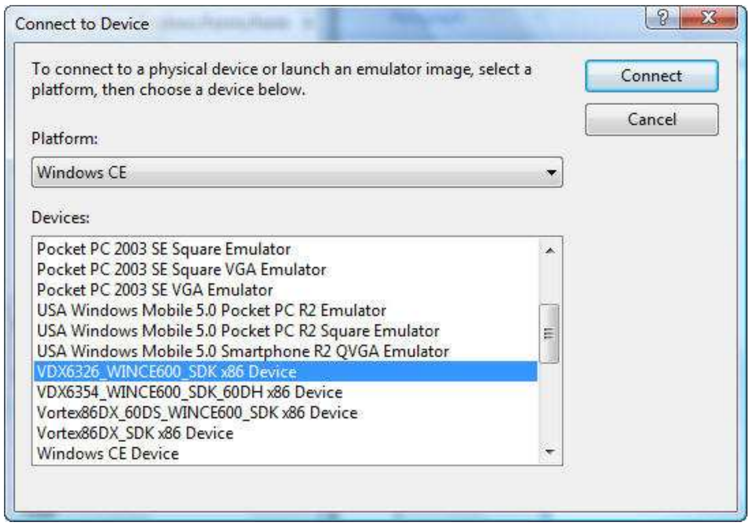
> The CoreCon_v200_x86 component is used for development purpose and should not be included in the release image, and should be removed from the image intended for distribution.

With the CoreCon_v200_x86 and AutoLaunch_v200_x86 components already included in the OS runtime image, along with the registry configured to launch CoreCon automatically when the CE 6.0 OS starts, the **Conmanclient2.exe** executable is launched automatically each time the OS runtime image starts.

> The following registry entries were added to the OS design by the CoreCon_v200_x86 component, to override system security and enable CoreCon connectivity to be established without the need to launch the **cMaccept.exe** executable:
>
> ```
> [HKEY_LOCAL_MACHINE\System]
>     "CoreConOverrideSecurity"=dword:1
> ```
>
> Without the above registry, in addition to ConmanClient2.exe, you need to launch cMaccept.exe executable to establish CoreCon connectivity.
>
> The cMaccept.exe executable needs to launch after ConmanClient2.exe. When executed, the cMaccept.exe disables the system security temporary to establish CoreCon connectivity, and will time-out in 3 minutes. If CoreCon connectivity is not established within 3 minutes, you need to launch cMaccept.exe again to establish CoreCon connectivity.

With ConmanClient2.exe automatically launched when the CE 6.0 OS runtime starts, work through the following steps to establish CoreCon connectivity between the development workstation's VS2005 IDE to the SBC:

- From VS2005 IDE, select **Tools | Connect to device…** and select **VDX6326_WINCE600_SDK** from the list of available devices, and click on the **Connect** button, as shown in Figure 43.



To connect to a physical device or launch an emulator image, select a platform, then choose a device below.

Platform:

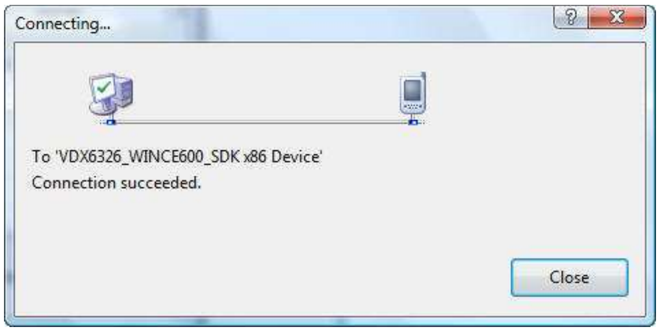
Windows CE

Devices:

Pocket PC 2003 SE Square Emulator
Pocket PC 2003 SE Square VGA Emulator
Pocket PC 2003 SE VGA Emulator
USA Windows Mobile 5.0 Pocket PC R2 Emulator
USA Windows Mobile 5.0 Pocket PC R2 Square Emulator
USA Windows Mobile 5.0 Smartphone R2 QVGA Emulator
VDX6326_WINCE600_SDK x86 Device
VDX6354_WINCE600_SDK_60DH x86 Device
Vortex86DX_60DS_WINCE600_SDK x86 Device
Vortex86DX_SDK x86 Device
Windows CE Device

Fig. 43 - Connect to Device (SBC)

When CoreCon connectivity is established, the Connecting dialog box will display Connection succeeded to indicate a successful connection, as shown in Figure 44.



To 'VDX6326_WINCE600_SDK x86 Device'
Connection succeeded.

Fig. 44 - Connected to the SBC

Step 4: Download C# Application to the VDX-6326 SBC

With the CoreCon Connectivity established, we are ready to download the C# application to the SBC.

From VS2005 IDE, select **Debug | Start Debugging** to deploy the application onto the SBC. From the VS2005 IDE's output window, a series of download activities are shown. After the necessary files are downloaded, the application is launched on the SBC's desktop, as shown on Figure 45.

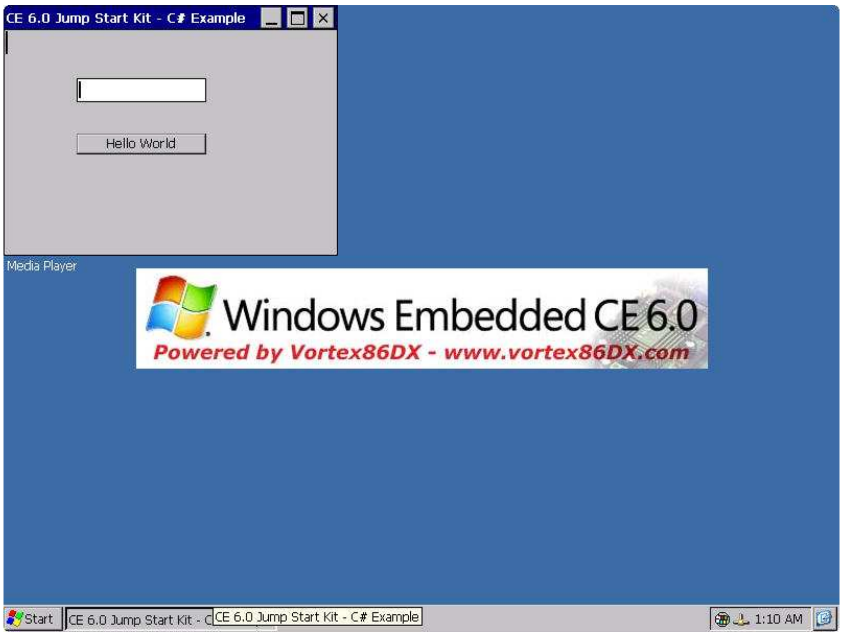

CE 6.0 Jump Start Kit - C# Example
Hello World
Media Player
Windows Embedded CE 6.0
Powered by Vortex86DX - www.vortex86DX.com
Start
CE 6.0 Jump Start Kit - C
CE 6.0 Jump Start Kit - C# Example
1:10 AM

Fig. 45 - CE 6.0 desktop with C# managed code application running

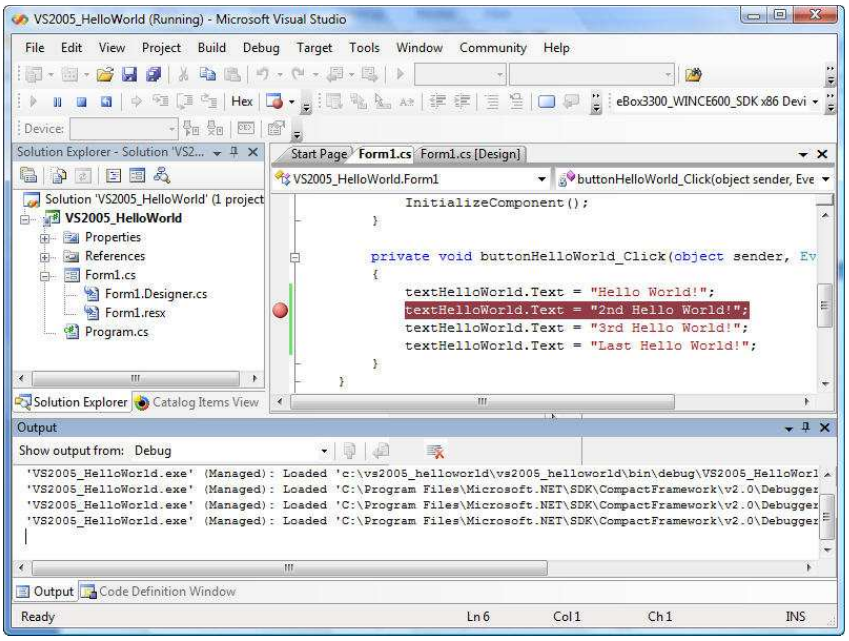
Step 5: Debug the C# Application running on the VDX-6326 SBC

While the VS2005_HelloWorld C# application is running on the SBC, we can set breakpoint to the source codes to halt the application when the application execution reaches the breakpoint.

Work through the following steps to set a breakpoint in the `buttonHelloWorld_Click` event:

- From the VS2005 IDE, navigate to the buttonHelloWorld_Click event code segment, click on the following line of code and press the F9 key to set a breakpoint, as shown in Figure 46.

```
textHelloWorld.Text = "2nd Hello World!";
```


VS2005_HelloWorld (Running) - Microsoft Visual Studio
File Edit View Project Build Debug Target Tools Window Community Help
eBox3300_WINCE600_SDK x86 Devi
Device:
Solution Explorer - Solution 'VS2...
Solution 'VS2005_HelloWorld' (1 project
VS2005_HelloWorld
Properties
References
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
Solution Explorer
Catalog Items View
Start Page
Form1.cs
Form1.cs [Design]
VS2005_HelloWorld.Form1
buttonHelloWorld_Click(object sender, Eve
Output
Show output from: Debug
Output
Code Definition Window
Ready
Ln 6
Col 1
Ch 1
INS

```
            InitializeComponent();
        }

        private void buttonHelloWorld_Click(object sender, Ev
        {
            textHelloWorld.Text = "Hello World!";
            textHelloWorld.Text = "2nd Hello World!";
            textHelloWorld.Text = "3rd Hello World!";
            textHelloWorld.Text = "Last Hello World!";
        }
    }
```

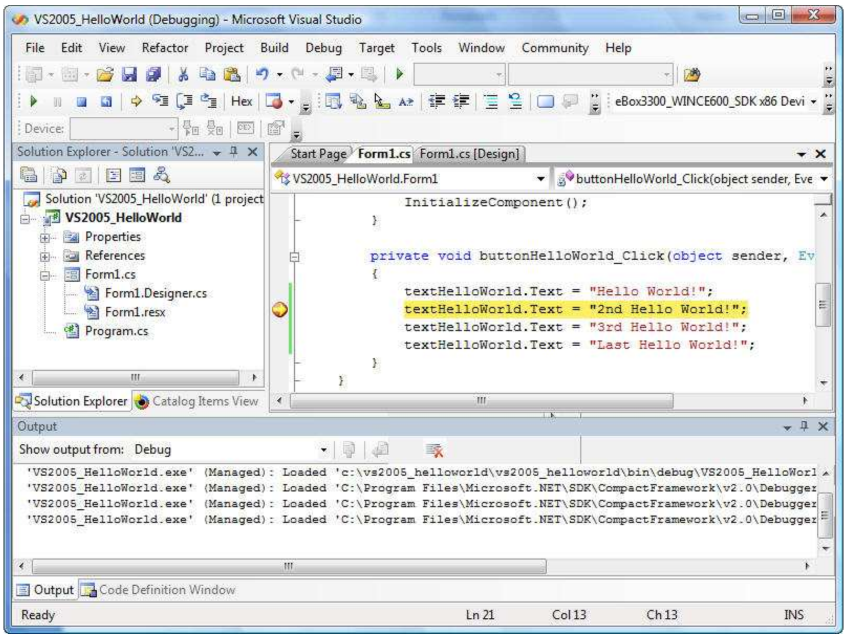
```
'VS2005_HelloWorld.exe' (Managed): Loaded 'c:\vs2005_helloworld\vs2005_helloworld\bin\debug\VS2005_HelloWorl
'VS2005_HelloWorld.exe' (Managed): Loaded 'C:\Program Files\Microsoft.NET\SDK\CompactFramework\v2.0\Debugger
'VS2005_HelloWorld.exe' (Managed): Loaded 'C:\Program Files\Microsoft.NET\SDK\CompactFramework\v2.0\Debugger
'VS2005_HelloWorld.exe' (Managed): Loaded 'C:\Program Files\Microsoft.NET\SDK\CompactFramework\v2.0\Debugger
```

Fig. 46 - Setting breakpoint

The above breakpoint is set while the VS2005_HelloWorld application is running on the SBC.

With the breakpoint in place, work through the following steps to execute the VS2005_HelloWorld application, reaching the breakpoint and halt.

- From the SBC CE 6.0 desktop, with the VS2005_HelloWorld application running, click on the **Hello World** button.
- From the VS2005 IDE on the development workstation, the breakpoint highlight color changed to **yellow**, an indication the program is halt at this line of code, as shown in Figure 47.


VS2005_HelloWorld (Debugging) - Microsoft Visual Studio
File Edit View Refactor Project Build Debug Target Tools Window Community Help
eBox3300_WINCE600_SDK x86 Devi
Solution Explorer - Solution 'VS2...
Solution 'VS2005_HelloWorld' (1 project
VS2005_HelloWorld
Properties
References
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
Solution Explorer
Catalog Items View
Start Page
Form1.cs
Form1.cs [Design]
VS2005_HelloWorld.Form1
buttonHelloWorld_Click(object sender, Eve
Output
Show output from: Debug
Output
Code Definition Window
Ready
Ln 21
Col 13
Ch 13
INS

```
            InitializeComponent();
        }

        private void buttonHelloWorld_Click(object sender, Ev
        {
            textHelloWorld.Text = "Hello World!";
            textHelloWorld.Text = "2nd Hello World!";
            textHelloWorld.Text = "3rd Hello World!";
            textHelloWorld.Text = "Last Hello World!";
        }
    }
```

```
'VS2005_HelloWorld.exe' (Managed): Loaded 'c:\vs2005_helloworld\vs2005_helloworld\bin\debug\VS2005_HelloWorl
'VS2005_HelloWorld.exe' (Managed): Loaded 'C:\Program Files\Microsoft.NET\SDK\CompactFramework\v2.0\Debugger
'VS2005_HelloWorld.exe' (Managed): Loaded 'C:\Program Files\Microsoft.NET\SDK\CompactFramework\v2.0\Debugger
'VS2005_HelloWorld.exe' (Managed): Loaded 'C:\Program Files\Microsoft.NET\SDK\CompactFramework\v2.0\Debugger
```

Fig. 47 - Execution halt at the breakpoint

- The application is running on the SBC and is halt on the following line of code.

  ```
  textHelloWorld.Text = "2nd Hello World!";
  ```

  As the code execution is halt on the above line of code, the code has not been executed. The textbox on the VS2005_HelloWorld application screen is showing the “Hello World!” message, indicating the line of code just before the breakpoint has been executed.

- From the VS2005 IDE, press the F11 key to step through one line of code.

  As we press the F11 key, we can see the next line of code becomes highlighted with yellow. The textbox on the VS2005_HelloWorld application running on the SBC is changed to “2nd Hello World!”.

- The F11 key is used to step through the code one line at a time. To continue the code execution, press the F5 key from the VS2005 IDE.

From the VS2005_HelloWorld application screen, running on the SBC, you can click on the **Hello World** button again to execute the code and reach the breakpoint to halt the execution.

As you can see from the simple exercise in this section, CE 6.0 and the Visual Studio development environment provide an effective, efficient and easy to use development environment.

You can develop VS2008 application using similar process.

Part 10 – Native Code Application with Visual Studio 2005

VS2005 and VS2008 can be used to develop native and managed code applications for CE 6.0.

In this section, we will cover the following:

- Develop a simple Win32 native code application using VS2005.
- Establish connection between the development workstation and SBC using CoreCon.
- Download the application to SBC with the CE 6.0 OS runtime image built in the earlier section.

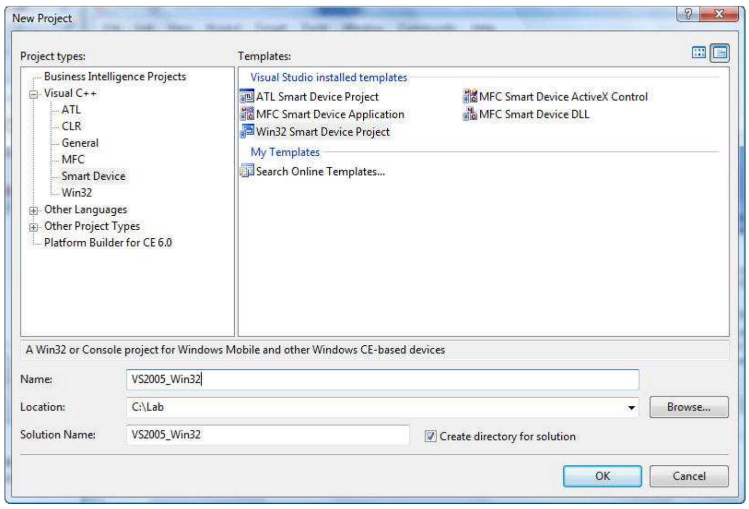
To work through the exercise in this section, you need to install the SDK generated from the OS design in the earlier section, VDX6326_WINCE600_SDK.msi.

You can use similar steps to develop application with VS2008.

> A native code application with Visual Studio 2008 exercise is provided on Appendix N, toward the end of this guide.
>
> Project files for the VS2005 and VS2008 native code exercises (VS2005_Win32 and VS2008_Win32) are provided in the \Application Sample\ folder on the jump start CD.

Step 1: Create a New Visual Studio 2005 C++ Project

From VS2005 IDE, Select **File | New | Project,** to bring up the New Project wizard, as shown in Figure 48.


New Project
Project types:
Business Intelligence Projects
Visual C++
ATL
CLR
General
MFC
Smart Device
Win32
Other Languages
Other Project Types
Platform Builder for CE 6.0
Templates:
Visual Studio installed templates
ATL Smart Device Project
MFC Smart Device ActiveX Control
MFC Smart Device Application
MFC Smart Device DLL
Win32 Smart Device Project
My Templates
Search Online Templates...
Browse...
OK
Cancel

A Win32 or Console project for Windows Mobile and other Windows CE-based devices

Name: VS2005_Win32

Location: C:\Lab

Solution Name: VS2005_Win32

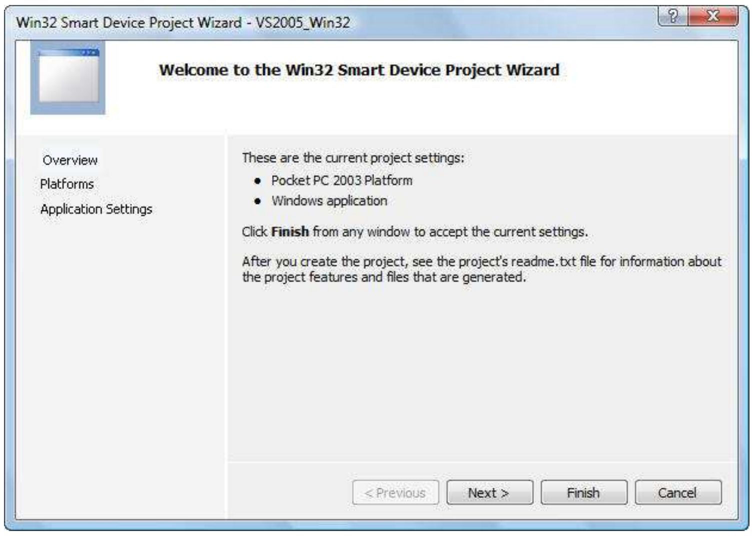
☑ Create directory for solution

Fig. 48 - VS2005 IDE / New Win32 native code Project

- On the new project wizard screen's left pane, expand **Visual C++** node and select **Smart Device** as the project type. On the right pane, select **Win32 Smart Device Project**.
- Enter **VS2005_Win32** as the name of the project.
- Enter **C:\Lab** as the location of the project.

You can choose a different folder to place the VS2005_Win32 project.

- Click on the **OK** button to bring up the Win32 Smart Device Project wizard, as shown in Figure 49.



Win32 Smart Device Project Wizard - VS2005_Win32

Welcome to the Win32 Smart Device Project Wizard

These are the current project settings:

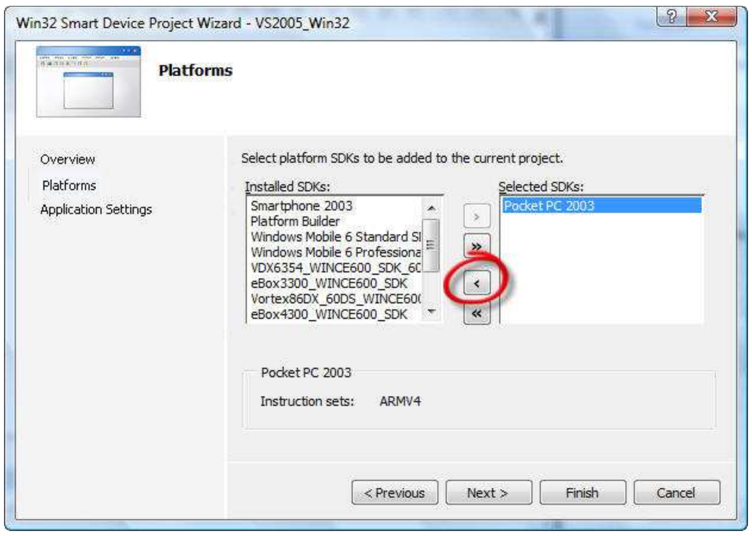
- Pocket PC 2003 Platform
- Windows application

Click **Finish** from any window to accept the current settings.

After you create the project, see the project's readme.txt file for information about the project features and files that are generated.

Fig. 49 - Win32 Smart Device Application wizard

- Click on the Next button to bring up the Platforms selection step, as shown in Figure 50.



Win32 Smart Device Project Wizard - VS2005_Win32

Platforms

Select platform SDKs to be added to the current project.

Installed SDKs:
Smartphone 2003
Platform Builder
Windows Mobile 6 Standard SI
Windows Mobile 6 Professiona
VDX6354_WINCE600_SDK_6C
eBox3300_WINCE600_SDK
Vortex86DX_60DS_WINCE60(
eBox4300_WINCE600_SDK

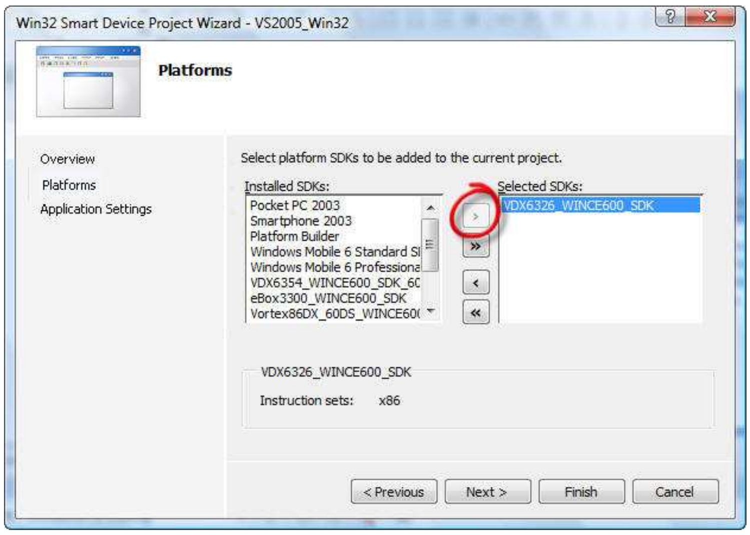
Selected SDKs:
Pocket PC 2003

Pocket PC 2003
Instruction sets: ARMV4

Fig. 50 - Platform and SDK selection

- From the Selected SDKs pane on the right, click to highlight the **PocketPC 2003** entry.
- Click on the single left pointing arrow, located to the left of the Selected SDKs pane to remove the **PocketPC 2003** entry from the selected SDKs pane.
- From the Installed SDKs pane on the left, click to highlight the **VDX6326_WINCE600_SDK** entry.

- Click on the right pointing arrow, located to the right of the Installed SDKs pane, to add the **VDX6326_WINCE600_SDK** to the selected SDKs pane, as shown in Figure 51.



Win32 Smart Device Project Wizard - VS2005_Win32

Platforms

Overview
Platforms
Application Settings

Select platform SDKs to be added to the current project.

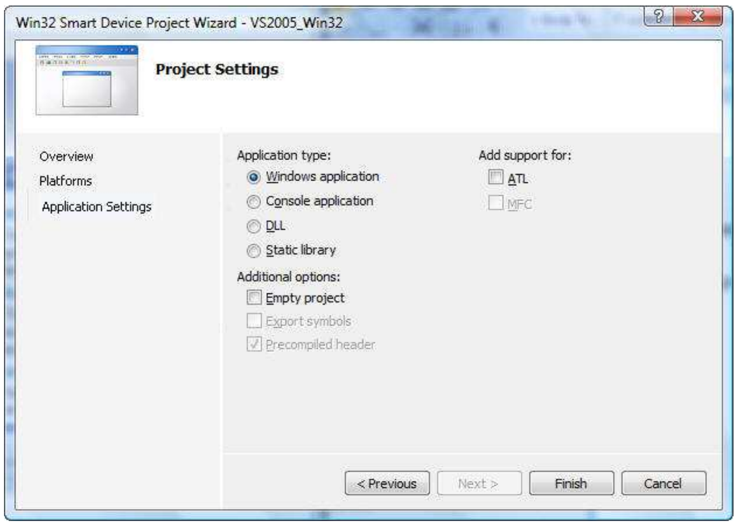
Installed SDKs:
Pocket PC 2003
Smartphone 2003
Platform Builder
Windows Mobile 6 Standard SI
Windows Mobile 6 Professiona
VDX6354_WINCE600_SDK_6C
eBox3300_WINCE600_SDK
Vortex86DX_60DS_WINCE60(

Selected SDKs:
VDX6326_WINCE600_SDK

VDX6326_WINCE600_SDK
Instruction sets: x86

Fig. 51 - Platform and SDK selection

- Click on the **Next** button to bring up the Project Settings step, as shown in Figure 52.



Win32 Smart Device Project Wizard - VS2005_Win32

Project Settings

Overview
Platforms
Application Settings

Application type:
(●) Windows application
( ) Console application
( ) DLL
( ) Static library

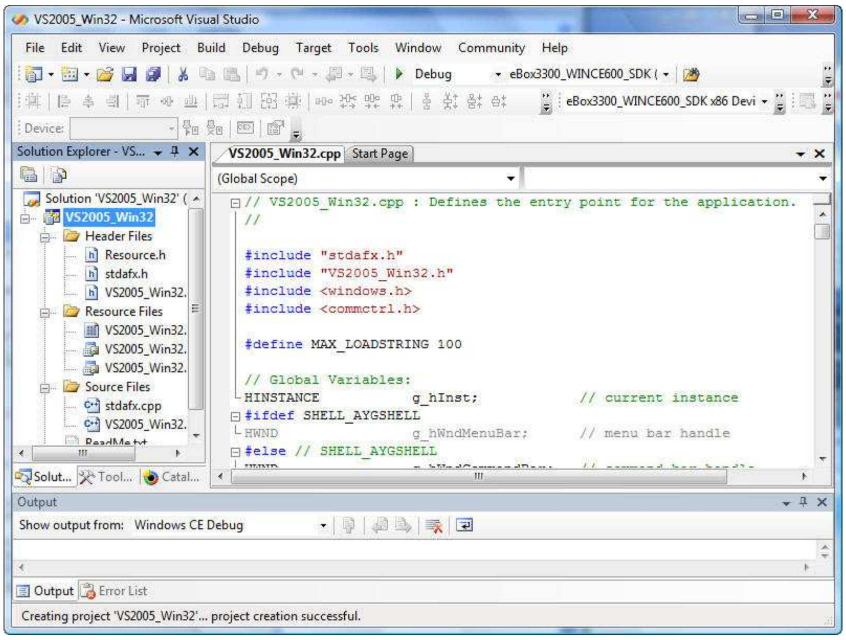
Additional options:
☐ Empty project
☐ Export symbols
☑ Precompiled header

Add support for:
☐ ATL
☐ MFC

Fig. 52 - Project settings

- Keep the default selection to create a **Windows application**.
- Click on the **Finish** button to complete the wizard and generate the initial project files.

After the project wizard step is completed, the VS2005_Win32 project files are created in the C:\Lab\ VS2005_Win32 folder. With the VS2005_Win32 project active, the VS2005 IDE should look similar to IDE as shown in Figure 53.



```
// VS2005_Win32.cpp : Defines the entry point for the application.
//

#include "stdafx.h"
#include "VS2005_Win32.h"
#include <windows.h>
#include <commctrl.h>

#define MAX_LOADSTRING 100

// Global Variables:
HINSTANCE           g_hInst;            // current instance
#ifdef SHELL_AYGSHELL
HWND                g_hWndMenuBar;      // menu bar handle
#else // SHELL_AYGSHELL
```

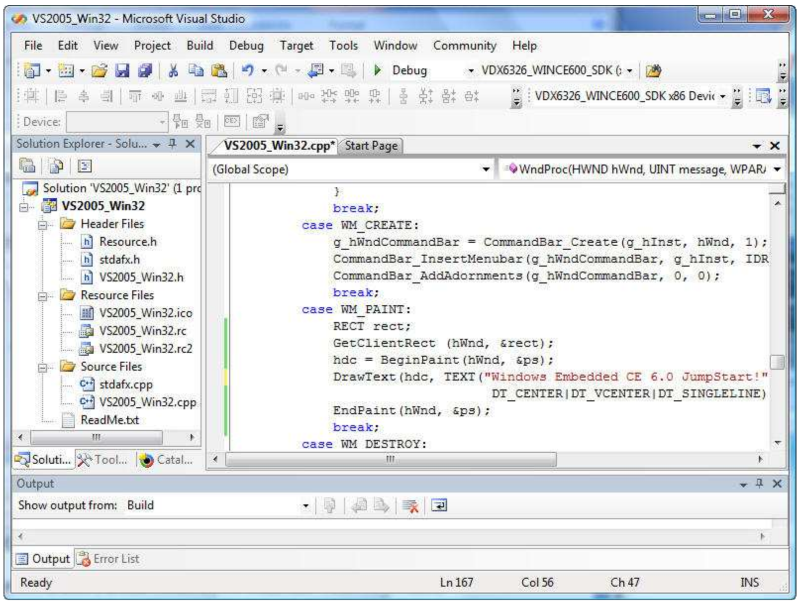
Creating project 'VS2005_Win32'... project creation successful.

Fig. 53 - VS2005 IDE with VS2005_Win32 project active

Work through the following steps to add some code to the VS2005_Win32 project.

- From the VS2005 IDE's Solution Explorer tab, double click on the **VS2005_Win32.cpp** source code file, in the `\Source` folder to view and edit the codes in the code editor window.
- Replace the codes in the "**case WM_PAINT:**" section with the following codes, as shown in Figure 54.

```
Case WM_PAINT:
    RECT rect;
    GetClientRect (hWnd, &rect);
    hdc = BeginPaint(hWnd, &ps);
    DrawText(hdc, TEXT("Windows Embedded CE 6.0 JumpStart!"),-1, &rect,
                                                    DT_CENTER|DT_VCENTER|DT_SINGLELINE);
    EndPaint(hWnd, &ps);
    break;
```



```
        }
        break;
    case WM_CREATE:
        g_hWndCommandBar = CommandBar_Create(g_hInst, hWnd, 1);
        CommandBar_InsertMenubar(g_hWndCommandBar, g_hInst, IDR
        CommandBar_AddAdornments(g_hWndCommandBar, 0, 0);
        break;
    case WM_PAINT:
        RECT rect;
        GetClientRect (hWnd, &rect);
        hdc = BeginPaint(hWnd, &ps);
        DrawText(hdc, TEXT("Windows Embedded CE 6.0 JumpStart!"
                           DT_CENTER|DT_VCENTER|DT_SINGLELINE)
        EndPaint(hWnd, &ps);
        break;
    case WM DESTROY:
```

VS2005_Win32 - Microsoft Visual Studio
File Edit View Project Build Debug Target Tools Window Community Help
Debug
VDX6326_WINCE600_SDK (:
VDX6326_WINCE600_SDK x86 Devic
Device:
Solution Explorer - Solu...
Solution 'VS2005_Win32' (1 pro
VS2005_Win32
Header Files
Resource.h
stdafx.h
VS2005_Win32.h
Resource Files
VS2005_Win32.ico
VS2005_Win32.rc
VS2005_Win32.rc2
Source Files
stdafx.cpp
VS2005_Win32.cpp
ReadMe.txt
Soluti... Tool... Catal...
VS2005_Win32.cpp* Start Page
(Global Scope)
WndProc(HWND hWnd, UINT message, WPAR/
Output
Show output from: Build
Output Error List
Ready Ln 167 Col 56 Ch 47 INS

Fig. 54 - VS2005 IDE with VS2005_Win32 project active.

- From the VS2005 IDE, select **Build | Build Solution** to build the VS2005_Win32 project.

Step 2: Preparing the SBC to Connect to Visual Studio 2005 IDE

To perform this portion of the exercise, CE 6.0 image configured and built during the previous sections of this guide must be downloaded and running on the SBC, and the VDX6326_WINCE600_SDK must be installed to the development workstation.

CoreCon is used to establish connectivity between the SBC and development workstation's VS2005 IDE.

To establish CoreCon connectivity, the following 5 CoreCon files need to be copied to the "\Windows" folder on SBC:

- Clientshutdown.exe
- ConmanClient2.exe
- CMaccept.exe
- eDbgTL.dll
- TcpConnectionA.dll

These files are installed to the following directory on the VS2005 development workstation, as part of the VS2005 installation:

```
\Program Files\Common Files\Microsoft Shared\CoreCon\1.0\Target\wce400\
```

CoreCon components supporting different type of processors are provided. There are multiple sub folders under the above directory, with name corresponding to supported processor. CoreCon files supporting the VDX-6326 SBC, built with x86 CPU, are in the \x86 sub folder.

There are different versions of CoreCon files, installed to the development workstation as part of the Visual Studio 2005 and Visual Studio 2008 installation.

CE 6.0 runtime image built with a different version of CoreCon files from the version on the development workstation will not be able to establish connectivity with the development workstation.

In the earlier section, during the customizing OS design steps, the CoreCon_v200_x86 component is added to the OS design. By adding this component, the same version of CoreCon files, installed to the development workstation as part of the Visual Studio installation, are included in the OS design and compiled as part of the CE 6.0 OS runtime image generated from the OS design.

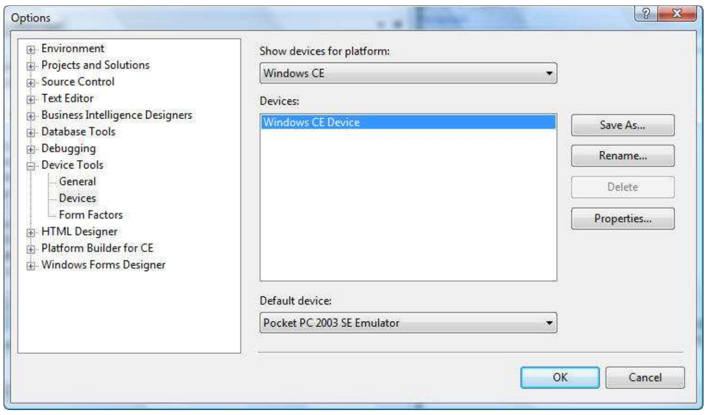
To establish connection between the VS2005 development workstation and the SBC, we need to know the SBC's IP address.

For this exercise, the SBC is connected to the development workstation using a cross-over RJ-45 Ethernet cable. The SBC is configured with a static IP address, 192.168.2.232 with 255.255.255.0 as the subnet. The development workstation is configured with a static IP address in the same subnet, 192.168.2.132.

Note: You can work through this exercise using static IP addresses with both the SBC and development workstation attached to an Ethernet hub without DHCP service.

Work through the following steps to establish connectivity between the SBC and VS2005 IDE:

- From VS2005 IDE, select **Tools | Options…** to bring up the following configuration screen, as shown in Figure 55.



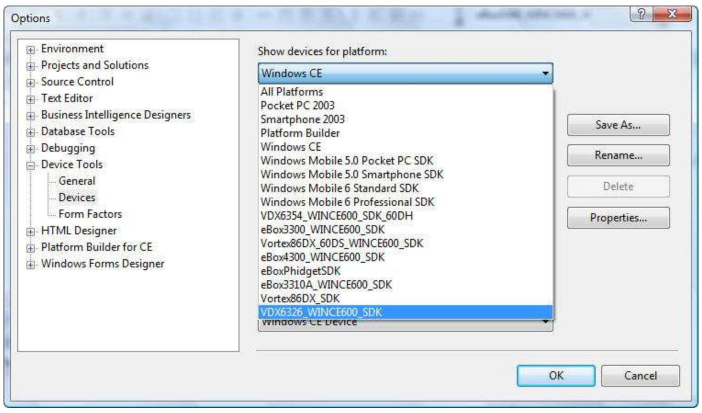
Show devices for platform: Windows CE

Devices: Windows CE Device

Default device: Pocket PC 2003 SE Emulator

Fig. 55 - VS2005 Tools Options

- On the left, click to expand the **Device Tools** node and select **Devices**.
- On the right, select **VDX6326_WINCE600_SDK** from the list of available platform in the **Show devices for platform** combo text box, as shown in Figure 56.

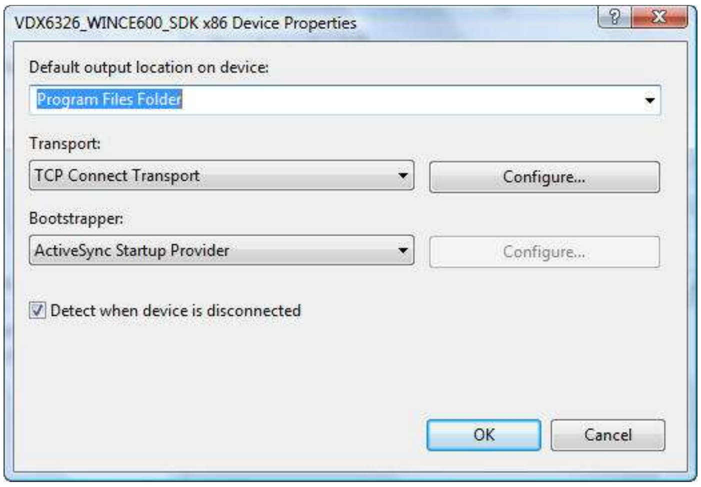


Show devices for platform: Windows CE

VDX6326_WINCE600_SDK

Fig. 56 - VS2005 Tools Options

- Click on the **Properties** button to bring up **VDX6326_WINCE600 x86 Device Properties** setting screen, as shown in Figure 57.



Default output location on device: Program Files Folder

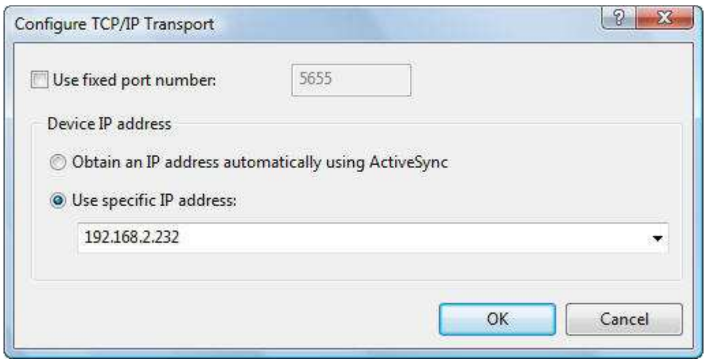
Transport: TCP Connect Transport

Bootstrapper: ActiveSync Startup Provider

☑ Detect when device is disconnected

Fig. 57 - VDX6326_WINCE600 x86 Device properties

- Click on the **Configure** button to bring up **Configure TCP/IP Transport** screen.
- Select **Use specific IP address** and enter IP address for the SBC, as shown in Figure 58.



☐ Use fixed port number: 5655

Device IP address

○ Obtain an IP address automatically using ActiveSync

● Use specific IP address: 192.168.2.232

Fig. 58 - Configure TCP/IP Transport / Set device IP Address

- Click the **OK** button to save device IP address setting.
- Click the **OK** button on the **VDX6326_WINCE600_SDK x86 Device Properties** screen.
- Click the **OK** button on the **Options** screen.

Step 3: Connecting SBC to Visual Studio 2005 IDE using CoreCon

To initiate connectivity between the development workstation's VS2005 IDE and SBC using CoreCon, we need to launch the **ConmanClient2.exe** and **cMaccept.exe** CoreCon components from the SBC, with CE 6.0 running. This is a cumbersome process and has to perform at least once for each time the SBC reset.

To make it easier to establish CoreCon connectivity, CoreCon files were added to the OS design and included to the OS runtime image. During customizing the OS design step, we added the CoreCon_v200_x86 component, AutoLaunch_v200_x86 component and entered registry entries to launch the CoreCon component automatically when the CE 6.0 OS starts.

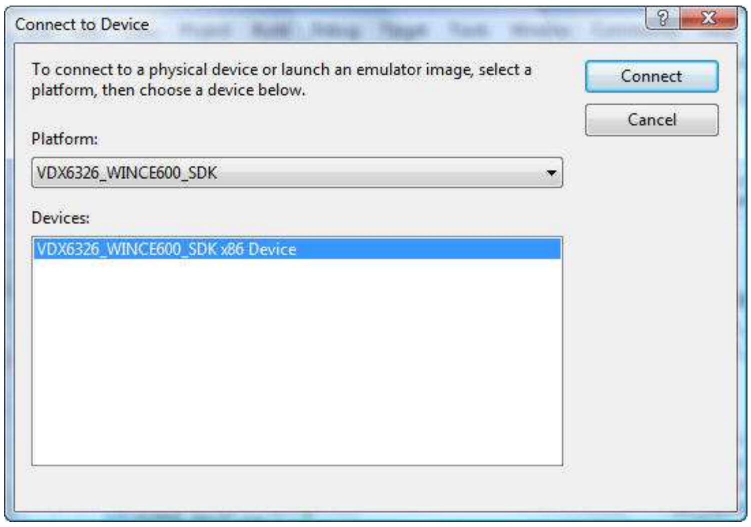
> The CoreCon_v200_x86 component is used for development purpose and should not be included in the release image, and should be removed from the image intended for distribution.

With the CoreCon_v200_x86 and AutoLaunch_v200_x86 components already included in the OS runtime image, along with the registry configured to launch CoreCon automatically when the CE 6.0 OS starts, the **Conmanclient2.exe** executable is launched automatically each time the OS runtime image starts.

> The following registry entries were added to the OS design by the CoreCon_v200_x86 component, to override system security and enable CoreCon connectivity to be established without the need to launch the **cMaccept.exe** executable:
>
> ```
> [HKEY_LOCAL_MACHINE\System]
>     "CoreConOverrideSecurity"=dword:1
> ```
>
> Without the above registry, in addition to ConmanClient2.exe, you need to launch cMaccept.exe executable to establish CoreCon connectivity.
>
> The cMaccept.exe executable needs to launch after ConmanClient2.exe. When executed, the cMaccept.exe disables the system security temporary to establish CoreCon connectivity, and will time-out in 3 minutes. If CoreCon connectivity is not established within 3 minutes, you need to launch cMaccept.exe again to establish CoreCon connectivity.

With ConmanClient2.exe automatically launched when the CE 6.0 OS runtime starts, work through the following steps to establish CoreCon connectivity between the development workstation's VS2005 IDE to the SBC:

- From VS2005 IDE, select **Tools | Connect to device…** and select **VDX6326_WINCE600_SDK** from the list of available devices, and click on the **Connect** button, as shown in Figure 59.


Connect to Device

To connect to a physical device or launch an emulator image, select a platform, then choose a device below.

Platform:

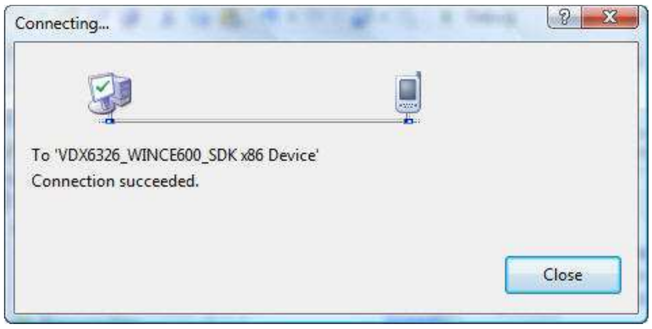
VDX6326_WINCE600_SDK

Devices:

VDX6326_WINCE600_SDK x86 Device

Fig. 59 - Connect to Device (SBC)

When CoreCon connectivity is established, the Connecting dialog box will display Connection succeeded to indicate a successful connection, as shown in Figure 60.


Connecting...

To 'VDX6326_WINCE600_SDK x86 Device'

Connection succeeded.

Fig. 60 - The SBC target device is connected

Step 4: Download VS2005_Win32 Application to the VDX-6326 SBC

With the CoreCon Connectivity established, we are ready to download the VS2005_Win32 application to the SBC.

From VS2005 IDE, select **Debug | Start Debugging** to deploy the **VS2005_Win32** application to the SBC.

After the VS2005_Win32 application is downloaded, it will launch on the SBC as shown in Figure 61.

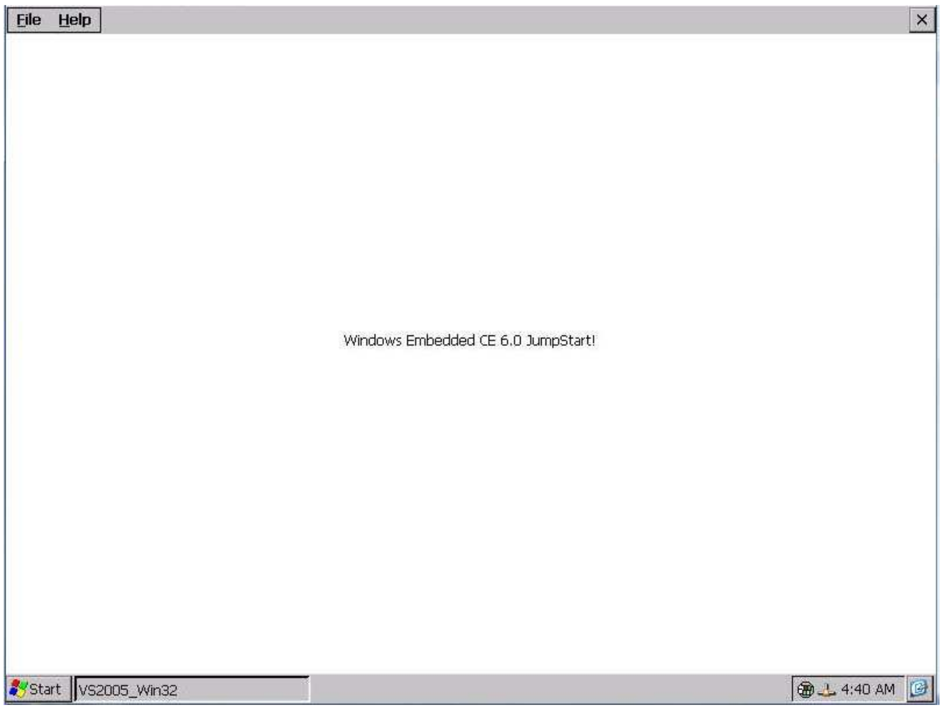


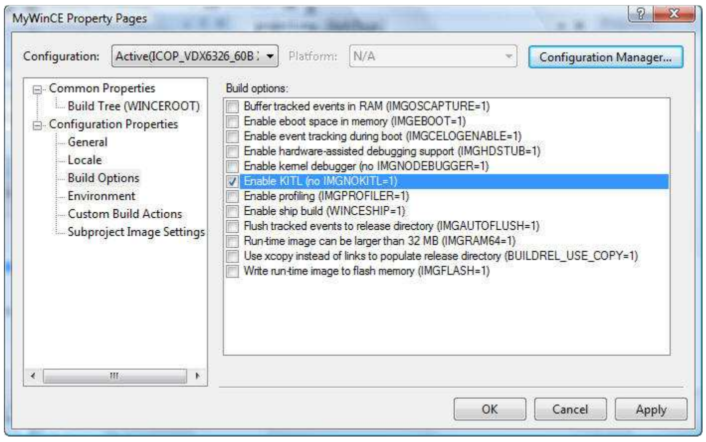
Fig. 61 - VS2005_Win32 application running on the SBC

Part 11 – Debug CE 6.0 OS Image with Remote Tools

While it’s not within the scope of this guide to cover in depth development and debug issues, we feel it’s informative to point out the remote tools and debug resources available from the VS2005 and Platform Builder IDE.

Using the same OS design project, MyWinCE, work through the following steps to enable the KITL (Kernel Independent Transport Layer) build option for the OS design and generate a new OS runtime image from the project.

- If the project is not already active, launch the MyWinCE OS design project created earlier.
- From the VS2005 IDE, select **Project | MyWinCE Properties…** to bring up the MyWinCE Property Pages screen.
- From the MyWinCE Property Pages screen’s left pane, expand the **Configuration Properties** node and click on the **Build Options** node to bring up the Build options pane on right, as shown in Figure 62.



MyWinCE Property Pages

Configuration: Active(ICOP_VDX6326_60B
Platform: N/A

Build options:
- [ ] Buffer tracked events in RAM (IMGOSCAPTURE=1)
- [ ] Enable eboot space in memory (IMGEBOOT=1)
- [ ] Enable event tracking during boot (IMGCELOGENABLE=1)
- [ ] Enable hardware-assisted debugging support (IMGHDSTUB=1)
- [ ] Enable kernel debugger (no IMGNODEBUGGER=1)
- [x] Enable KITL (no IMGNOKITL=1)
- [ ] Enable profiling (IMGPROFILER=1)
- [ ] Enable ship build (WINCESHIP=1)
- [ ] Flush tracked events to release directory (IMGAUTOFLUSH=1)
- [ ] Run-time image can be larger than 32 MB (IMGRAM64=1)
- [ ] Use xcopy instead of links to populate release directory (BUILDREL_USE_COPY=1)
- [ ] Write run-time image to flash memory (IMGFLASH=1)

Fig. 62 - MyWinCE Property Pages - Build options

- On the right Build options pane, select the **Enable KITL (no IMGNOKITL=1)** option.
- Click on the **Apply** button following by the **OK** button to save the setting and close the screen.
- From the VS2005 IDE, select **Build | Advanced Build Commands | Build Current BSP and Subprojects** to generate the OS runtime image with KITL.

Since the OS design has been SYSGEN in the earlier steps, and the build option configuration change does not require SYSGEN, we can use the advanced build command to save time.

Although the image is built in release mode, with the KITL build option enabled, we can use the remote tools to debug the image.

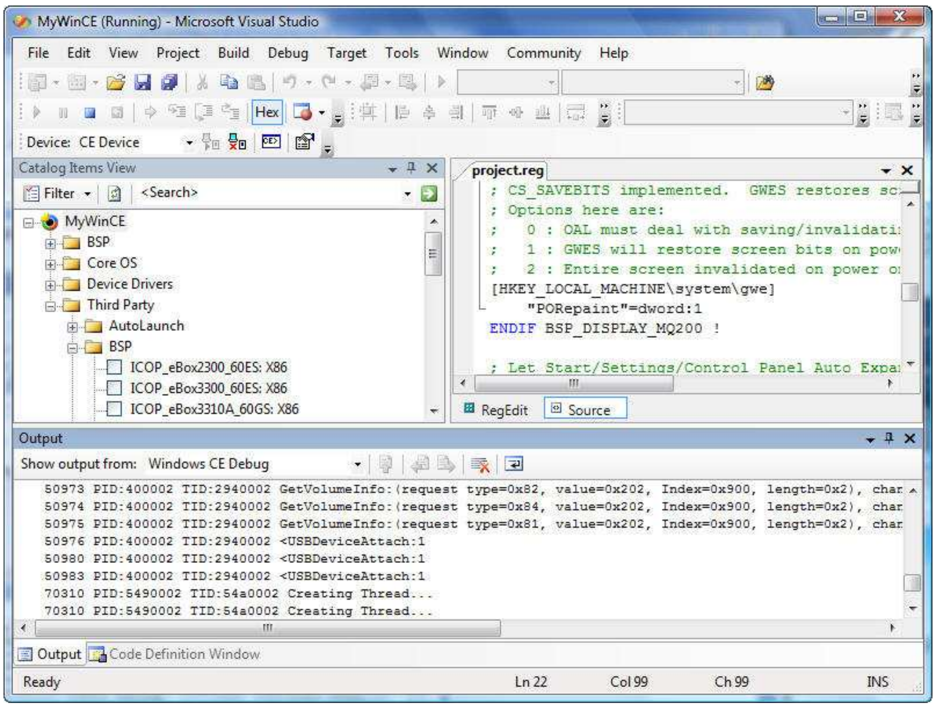
A debug mode image will provide far more debug information, needed for low level debug.

For most of the general application development environment, a release mode image with KITL enabled provides sufficient debug information.

Download CE 6.0 Image with KITL to the VDX-6326 SBC

After the build process in the previous step is completed and successfully generated an OS runtime image with KITL enabled. Follow the procedure in the earlier section, "Part 7 - Download OS Image to Target Device", to download the runtime image to the SBC.

After the image is downloaded to the SBC, the Platform Builder IDE should look similar to the following screen, as shown in Figure 63.



```
; CS_SAVEBITS implemented.  GWES restores sc
; Options here are:
;   0 : OAL must deal with saving/invalidati
;   1 : GWES will restore screen bits on pow
;   2 : Entire screen invalidated on power o
[HKEY_LOCAL_MACHINE\system\gwe]
    "PORepaint"=dword:1
ENDIF BSP_DISPLAY_MQ200 !

; Let Start/Settings/Control Panel Auto Expa
```

```
50973 PID:400002 TID:2940002 GetVolumeInfo:(request type=0x82, value=0x202, Index=0x900, length=0x2), char
50974 PID:400002 TID:2940002 GetVolumeInfo:(request type=0x84, value=0x202, Index=0x900, length=0x2), char
50975 PID:400002 TID:2940002 GetVolumeInfo:(request type=0x81, value=0x202, Index=0x900, length=0x2), char
50976 PID:400002 TID:2940002 <USBDeviceAttach:1
50980 PID:400002 TID:2940002 <USBDeviceAttach:1
50983 PID:400002 TID:2940002 <USBDeviceAttach:1
70310 PID:5490002 TID:54a0002 Creating Thread...
70310 PID:5490002 TID:54a0002 Creating Thread...
```

Fig. 63 - VS2005 IDE showing debug output message after image is downloaded to the SBC

With KITL enabled, the Platform Builder's debug output window displays additional information during the SBC boot process. From the debug output window, it shows the library and driver components' loading processes. The additional information is useful, and may be critical, to help debug, solve problems and prevent potential problems. Debug output message can be copied to a text file to be analyze in detail. Refer to Platform Builder online document for more information about various debug and troubleshooting resources available.

To see more detail debug information, in addition to KITL, enable Kernel Debugger build option.

To see even more detail debug information, in addition to enabling the KITL and Kernel Debugger build options, configure the OS design to generate debug mode image.

Next, we will work through a series of exercise using the Platform Builder remote tools to access the CE 6.0 OS runtime on the SBC remotely.

Remote Tools: Process Viewer

The Remote Process Viewer enables you to view the CE 6.0 OS' running processes, associated threads and modules running on the SBC.

To use this remote tool, with the OS image, built with KITL option, downloaded to the SBC, select **Target | Remote Tools | Process Viewer** from the VS2005 IDE to bring up the following screen, the **Select a Windows CE Device** screen, as shown in Figure 64.

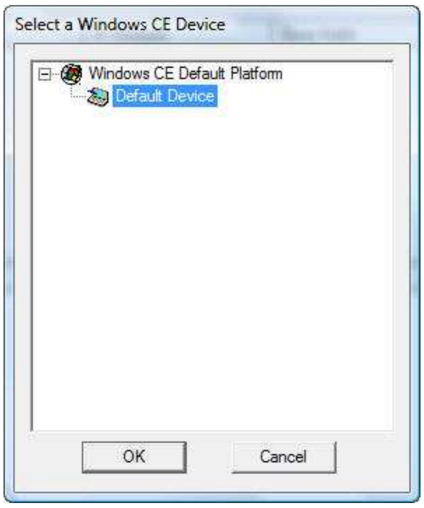
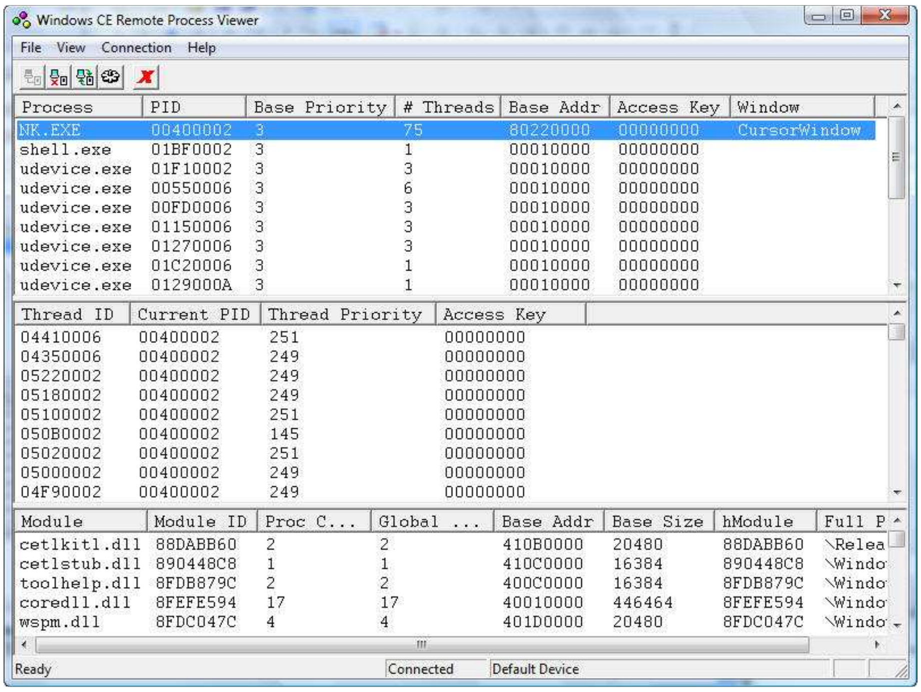


Fig. 64 - Select a Windows CE device to connect to the remote tool

Select the **Default Device** and click on **OK** to continue.

After the connection is established, the following screen, the **Windows CE Remote Process Viewer** screen, will appear, as shown in Figure 65.



Process	PID	Base Priority	# Threads	Base Addr	Access Key	Window
NK.EXE	00400002	3	75	80220000	00000000	CursorWindow
shell.exe	01BF0002	3	1	00010000	00000000	
udevice.exe	01F10002	3	3	00010000	00000000	
udevice.exe	00550006	3	6	00010000	00000000	
udevice.exe	00FD0006	3	3	00010000	00000000	
udevice.exe	01150006	3	3	00010000	00000000	
udevice.exe	01270006	3	3	00010000	00000000	
udevice.exe	01C20006	3	1	00010000	00000000	
udevice.exe	0129000A	3	1	00010000	00000000	

Thread ID	Current PID	Thread Priority	Access Key
04410006	00400002	251	00000000
04350006	00400002	249	00000000
05220002	00400002	249	00000000
05180002	00400002	249	00000000
05100002	00400002	251	00000000
050B0002	00400002	145	00000000
05020002	00400002	251	00000000
05000002	00400002	249	00000000
04F90002	00400002	249	00000000

Module	Module ID	Proc C...	Global ...	Base Addr	Base Size	hModule	Full P
cetlkitl.dll	88DABB60	2	2	410B0000	20480	88DABB60	\Relea
cetlstub.dll	890448C8	1	1	410C0000	16384	890448C8	\Windo
toolhelp.dll	8FDB879C	2	2	400C0000	16384	8FDB879C	\Windo
coredll.dll	8FEFE594	17	17	40010000	446464	8FEFE594	\Windo
wspm.dll	8FDC047C	4	4	401D0000	20480	8FDC047C	\Windo

Fig. 65 - Select a Windows CE device to connect to the remote tool

Using the Remote Process Viewer, you can view all of the running processes and each process' associated threads and module. You can use the Remote Process Viewer to terminate running process. Try the following exercise.

Launch the Windows Media Player application on the SBC. From the Remote Process Viewer menu, select **Connection | Refresh**. The ceplayer.exe process will shows up on the Process window. Click on the ceplayer.exe to highlight this process. From the Remote Process Viewer menu, select **File | Terminate Process**. The Windows Media Player application running on SBC will terminated.

From the Remote Process Viewer menu, select **File | Exit** to terminate the remote tool session.

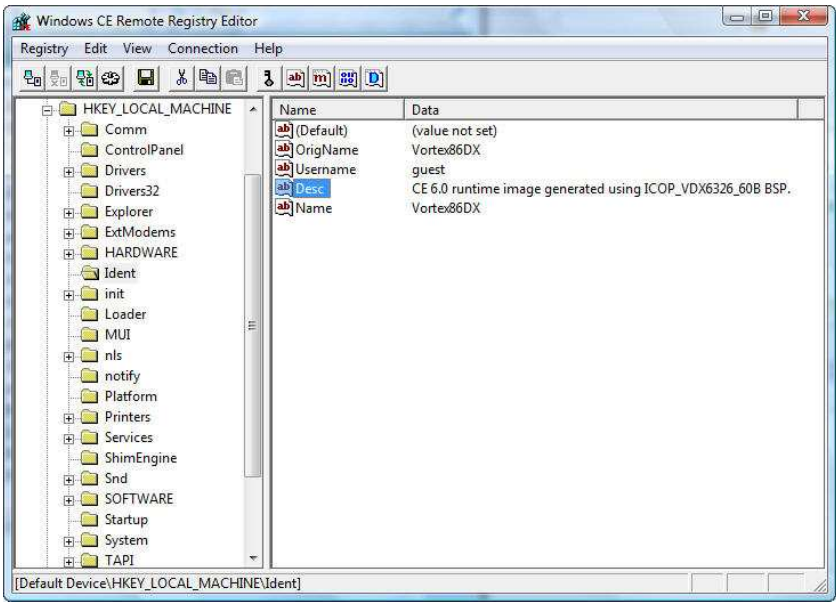
Remote Tools: Registry Editor

The Remote Registry Editor enables you to view and edit CE 6.0 registry entries on the SBC, from the development workstation.

To use this remote tool, with the OS image, built with KITL and Kernel Debugger options, downloaded to SBC, work through the following steps to launch the Remote Registry Editor:

- From the VS2005 IDE, select **Target | Remote Tools | Registry Editor** to bring up the **Select a Windows CE Device** screen.
- From the **Select a Windows CE Device** screen, click on the **Default Device** selection follow by the OK button to establish connection to the SBC.

After the remote connection is established, the following **Windows CE Remote Registry Editor** screen will appear, as shown in Figure 66.


Windows CE Remote Registry Editor
Registry Edit View Connection Help

HKEY_LOCAL_MACHINE
- Comm
- ControlPanel
- Drivers
- Drivers32
- Explorer
- ExtModems
- HARDWARE
- Ident
- init
- Loader
- MUI
- nls
- notify
- Platform
- Printers
- Services
- ShimEngine
- Snd
- SOFTWARE
- Startup
- System
- TAPI

Name	Data
(Default)	(value not set)
OrigName	Vortex86DX
Username	guest
Desc	CE 6.0 runtime image generated using ICOP_VDX6326_60B BSP.
Name	Vortex86DX

[Default Device\HKEY_LOCAL_MACHINE\Ident]

Fig. 66 - Remote Registry Editor

The Remote Registry Editor allows you to view and edit CE 6.0 registry entries remotely. This is a useful tool to view registry entries and check whether the registry entries for an associated device or application are included to the OS runtime.

You can also view which device drivers are loaded, by reviewing the registry entries under the following key.

```
[HKEY_LOCAL_MACHINE\Drivers\Active]
```

From the Windows CE Remote Registry Editor menu, select **Registry | Exit** to terminate the Remote Registry Editor tool.

Note: Using similar steps, you can access and try out the other Remote Tools.

Summary

You have now completed all the steps in this guide. Here's what we have covered:

- Develop a CE 6.0 OS design project.
- Configure and customize the OS design.
- Generate CE 6.0 OS runtime image from the OS design.
- Download the CE 6.0 OS runtime image to the VDX-6326 target device.
- Develop C# managed code application using VS2005, establish CoreCon connectivity and deploy the managed code application to the CE 6.0 OS image on the VDX-6326 target device.
- Develop native code application using VS2005, establish CoreCon connectivity and deploy the native code application to the CE 6.0 OS image on the VDX-6326 target device
- Configure the OS design with KITL to generate a CE 6.0 OS runtime with KITL enabled, download the OS runtime to the VDX-6326 target device and use remote tools to debug the image remotely.

This is just the beginning. Working on an embedded development project is quite different from developing desktop and enterprise application. Depending on the type of device you are working with, you need to have accurate information about the hardware you are working with and the appropriate BSP and hardware support library.

With the right hardware and Board-Support-Package, the Windows Embedded CE and the Visual Studio development IDE provide an efficient and effective development environment to help simplify difficult development tasks.

Visit the following web sites to learn more about hardware and other Windows Embedded technical resources:

http://www.icoptech.com

http://www.embeddedpc.net

http://www.embedded101.com

Note: Additional information resources are provided in the appendixes. Please take time and review the information in the appendixes.

Congratulations! - You've completed all the steps.

Here are some of the additional resources to help you gain Windows Embedded CE knowledge.

Microsoft Web sites.

Windows Embedded CE on MSDN
http://msdn.microsoft.com/en-us/windowsembedded/ce/default.aspx

Learn Windows Embedded CE
http://msdn.microsoft.com/en-us/windowsembedded/ce/dd367860.aspx

Windows Embedded News groups and Community
http://msdn.microsoft.com/en-us/windowsembedded/ce/dd424664.aspx

To learn more about the VDX-6326 SBC and other ICOP hardware, visit the following Web sites.

http://www.icoptech.com

http://www.embeddedpc.net

Or contact ICOP email: wep@icoptech.com

Phone: (626) 444-6666

Update to this jump start guide and other Windows Embedded CE information resources are available at the following web site:

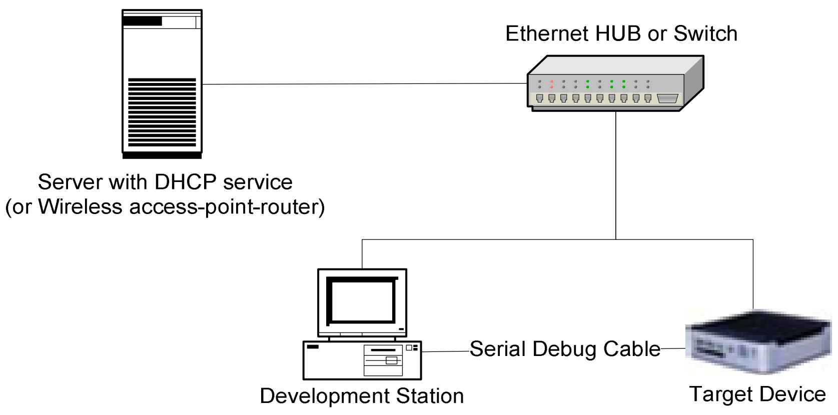
http://www.embeddedpc.net

http://www.embeddedpc.net/KnowledgeBase/

Additional information references are available in the Appendixes.

Appendix A – Development Environment with DHCP service

Both the development workstation and SBC are connected to the same Local-Area-Network with DHCP service.



Connecting to Local Area Network with DHCP

It's a typical setup to connect both the development workstation and the target device to a Local Area Network with DHCP service to provide IP addresses dynamically.

If the target device fails to establish connectivity with the development workstation and download the image as expected with this configuration, you may need to enable DHCP service for the target device on your network. Some secured network may require the target device's MAC address to be added to the authorized device list for DHCP service in the DHCP server.

Using Wireless Access Point Router

When using a wireless-access-point-router with multiple Ethernet ports, connecting both the development workstation and target device directly to the Ethernet port on the wireless-access-point-router may cause problem and prevent the development environment to function as expected.

Note: The wireless-access-point-router device's routing function filter and route network packets based on the packets' associated origin and destination information. The router may not route all of the network packets between the development workstation and the target device.

Instead of connecting directly to the wireless-access-point-router device's Ethernet ports, attach an Ethernet network hub or switch to the wireless-access-point-router device, to access the DHCP service, and connect both the development workstation and target device to the Ethernet network hub or switch.

Capturing Serial Debug Messages

The null serial debug cable is connected between one of the development workstation's serial ports and the target device's COM1 to capture serial debug messages from the target device.

By connecting a null serial modem cable between one of the serial ports on the development workstation and the target device's COM1 and configure the serial port to 38400-8-N-1, HyperTerminal can be launched on the development workstation to view debug messages output from the target device's serial port.

Note: If you are using Windows Vista workstation, Hyper Terminal is no longer available as part of the OS. Refer to Appendix Q for information about Hyper Terminal for Windows Vista.

Appendix B – Development Environment Without DHCP – Static IP Addresses

The information in this section applies to the following two scenarios:

1. A cross-over RJ-45 Ethernet cable is connected between the target device's Ethernet port and the development workstation's Ethernet port, as shown in Figure B1.

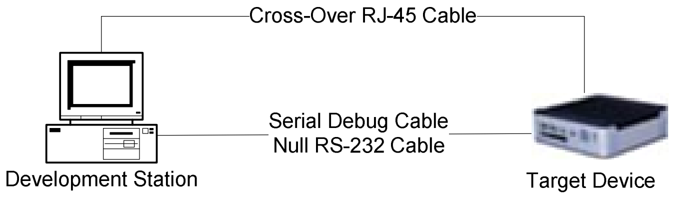


Fig. B1

2. The target device and the development workstation are attached to the same Ethernet hub or switch without DHCP service, as shown in Figure B2.

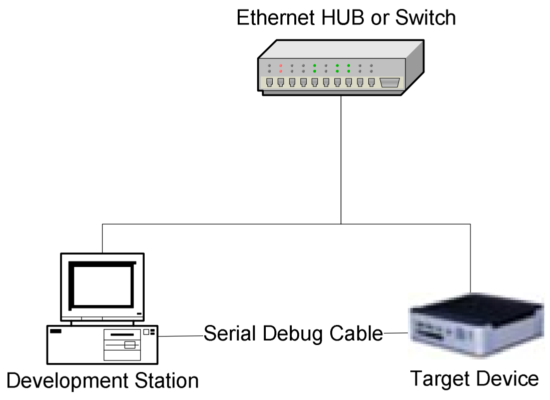


Fig. B2

Static IP Address

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

When using the Windows Embedded CE 6.0 jump start kit's preconfigured software with static IP address, 192.168.2.232, with 255.255.255.0 subnet mask, is the static IP address pre-configured for target device.

To establish connectivity to the target device, the development workstation must be configured with a static IP address in same subnet, as follow:

IP address:	192.168.2.xxx (xxx can be any value ranging from 1 to 255, except 232)
Subnet mask:	255.255.255.0

The prebuilt CE 6.0 OS runtime images provided as part of the preconfigured software on the target device's local flash storage is built with DHCP enabled, and will attempt to send request to acquire IP address from an available DHCP server during startup. Refer to Appendix-H for information about how to configure a static IP address for the CE 6.0 OS runtime, and how to modify the OS design's registry entries to configure and build a CE 6.0 OS runtime with static IP address.

Note: If the IP address is not setup correctly, the target device will not able to communicate with the development workstation.

Capturing Serial Debug Messages

The null serial debug cable is connected between one of the development workstation's serial ports and the target device's COM1 to capture serial debug messages from the target device.

By connecting a null serial modem cable between one of the serial ports on the development workstation and the target device's COM1 and configure the serial port to 38400-8-N-1, HyperTerminal can be launched on the development workstation to view debug messages output from the target device's serial port.

Note: If you are using Windows Vista workstation, Hyper Terminal is no longer available as part of the OS. Refer to Appendix Q for information about Hyper Terminal for Windows Vista.

Appendix C – Useful information for Windows Embedded CE

Windows CE Reference

Windows Embedded CE information resources on MSDN
http://msdn.microsoft.com/en-us/windowsembedded/ce/default.aspx
This is the main landing page for Windows Embedded CE on MSDN. If you are new to Windows Embedded CE, this site provide valuable information and links to Windows Embedded CE resources.

Windows Embedded Tutorials
http://msdn.microsoft.com/en-us/windowsembedded/ce/dd367860.aspx
This site provides how-to tutorials on basic and advanced topics about using Windows CE in development of embedded devices.

Windows Embedded News Group
http://msdn.microsoft.com/en-us/windowsembedded/ce/dd424664.aspx

Windows Embedded CE Community projects

Open SSH for Windows CE
http://www.codeplex.com/wiki/view.aspx?projectname=CESSH
Remotely access a Windows CE device in a secure manner using the SSH protocol. It helps execute remote commands on the device, but it also supports tunneling, forwarding arbitrary TCP ports and it can transfer files using the associated SFTP or SCP protocols.

Windows CE Wifi Driver for Atheros AR-6000
http://www.codeplex.com/wiki/view.aspx?projectname=CEWifiDriverAR6000
This driver helps connect the Atheros AR-6000 Wifi chipset to your Windows CE device.

32feet.Net – Personal Area Networking for .NET
http://www.codeplex.com/32feet
32feet.NET is a project to make personal area networking technologies such as Bluetooth, Infrared (IrDA) and more, easily accessible from .NET code. Requires .NET Compact Framework v1.0 or above and Windows CE.NET 4.2 or above, or .NET Framework v1.1 for desktop Windows XP.

Phidgets USB I/O driver shared source projects
http://www.codeplex.com/PhidgetsWinCEDriver
Phidgets are an easy to use set of building blocks for low cost sensors and controllers. This community project is a driver to allow libraries and applications access to USB Phidgets.

Bluetooth Wrapper for Windows CE
http://msdn2.microsoft.com/en-us/embedded/aa714519.aspx
Provides a free Win32 API Wrapper that developers can expose in Visual Studio .NET or the .NET Compact Framework. Exposing the Win32 API Wrapper reduces the amount of code needed to develop for Bluetooth Technologies and helps make it easier to create compelling Windows Mobile and Windows CE Bluetooth applications.

LSP Samples for Windows CE
http://www.codeplex.com/LSPSamplesWindowsCE
This project provides the code necessary to create LSPs (Layered Service Providers) on Windows CE and Windows Mobile.

USB Webcam Driver for Windows CE
http://www.codeplex.com/cewebcam

This project provides Windows CE device driver to support USB Webcam built to meet the USB video class specification.

Windows Embedded CE 6.0 USB Camera Driver
http://www.microsoft.com/downloads/details.aspx?FamilyID=2ef087c0-a4ae-42cc-abd0-c466787c11f2&DisplayLang=en
This is the URL to the USB camera driver for Windows Embedded CE 6.0 to support USB camera built to the USB video class specification.

Other Useful Links

http://www.embedded101.com

http://www.learningce.com

http://www.embeddedpc.net

Hardware Reference Information

http://www.icoptech.com
The VDX-6326 SBC is designed with the Vortex86DX System-On-Chip, with 256MB DDR2 system memory.

Appendix D – VDX-6326 Single-Board-Computer Technical Information

	Features **Fan-less Design** **800 MHz Vortex86DX** **XGI Z9s video with VGA and LCD support** **256MB DDR2 RAM** **4MB SPI Flash** **3 10/100Mbps Ethernet** **UltraDMA-100/66/33 IDE** **CM119 Audio** **3 USB 2.0 host interfaces** **3 RS-232 serial and 1 parallel ports** **1 RS-232/422/485 serial port** **CompactFlashSlot** **Mini-PCI expansion slot** **PC/104 expansion slot**

The VDX-6326 Single-Board-Computer (SBC) is an industrial SBC, designed with low-power consumption to operate without fan using passive cooling, and able to operate in harsh temperature range from -20℃ to +70℃. Industrial grade versio n, able to operate from -40℃ to +85℃, is availabl e by special order.

Designed with the ultra low power 800 MHz Vortex86DX System-On-Chip with integrated I/O peripherals and soldered on 256MB DDR2 RAM provides sufficient system memory to support Windows Embedded CE, Windows XP Embedded and WEPOS. The 3 10/100Mbps Ethernet, the 3 high bandwidth USB 2.0 ports, the 4 serial ports, Mini-PCI slot and PC/104 interface provide the interfaces to connect to broad range of external devices and systems with endless application possibility.

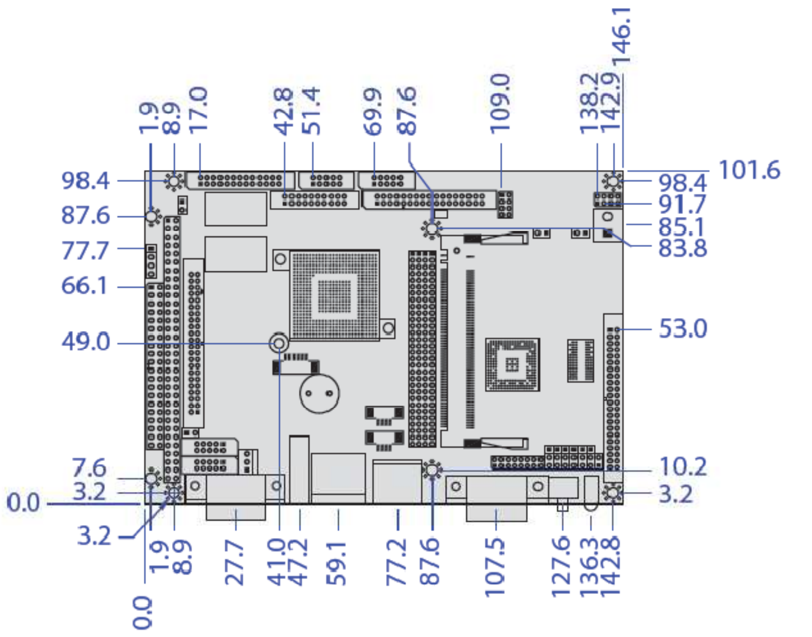
The VDX-6326 SBC supports multiple boot options and able to boot from the following resources:

- Internal IDE storage
- Compact Flash storage
- USB floppy
- USB storage
- USB CD/DVD-ROM
- Onboard SPI flash
- Boot from Network via PXE

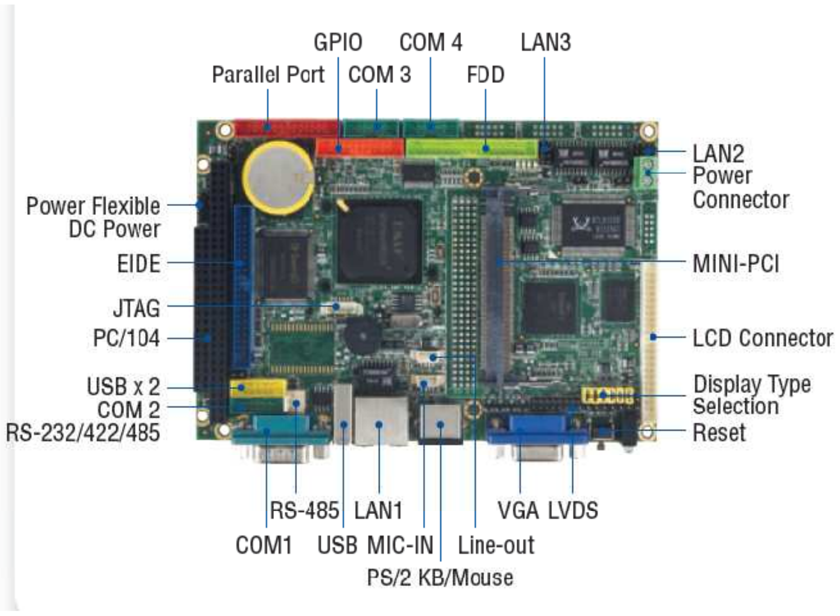
Board-Support-Package is available to support Windows Embedded CE 5.0, CE 6.0, CE 6.0 R2 and CE 6.0 R3. Device drivers are available to support Windows XP, Windows XP Embedded and WEPOS.

Designed with the matured x86 CPU architecture with rich set of integrated I/O and able to operate in harsh environment, the VDX-6326 SBC is the suitable for broad range of application, such as industrial-automation, process-control, robotics, instrumentation, human-machine-interface, medical device, automotive, utility metering, network appliance, security access control, thin client, intelligent RFID reader, home-building automation, point-of-sales, information kiosk and other embedded devices.

By integrating key peripherals into a product quality design, along with the support for the Windows Embedded technologies, the VDX-6326 hardware platform can help developer minimize development risk, significantly lower development cost and gain critical Time-to-Market advantage.



VDX-6326 SBC dimension



VDX-6326 SBC I/O peripherals

VDX-6326 SBC Specification

System

CPU	800 MHz Vortex86DX System-On-Chip
BIOS	AMI BIOS
System Chipset	Integrated in Vortex86DX
I/O Chip	Integrated in Vortex86DX
System Memory (RAM)	Soldered on 256MB DDR2 RAM
Storage	Compact Flash, Internal IDE and SPI Flash

I/O

	1 x EIDE (UltraDMA 133) 1 x VGA 1 x LCD (TFT) 1 x LVDS 3 x USB 3 x LAN 1 x Mic. In 1 x Line out 1 x Type I/II CF Slot 1 x PS/2 K/B & Mouse 3 x RS-232 1 x RS-232/422/485 1 x Parallel 1 x Mini-PCI socket 1 x PC/104 expansion 1 x 16-bit GPIO

Display

Chipset	XGI Z9s
Display Memory	32MB DDR2 memory
Display Resolution	Up to 1600 x 1200

Audio

Chipset	CM119
Audio Interface	Mic-in, Line-out

Ethernet

Chipset	10/100Mbps R6040, integrated in Vortex86DX
	2 x 10/100Mbps Realtek-8100

Mechanical & Environment

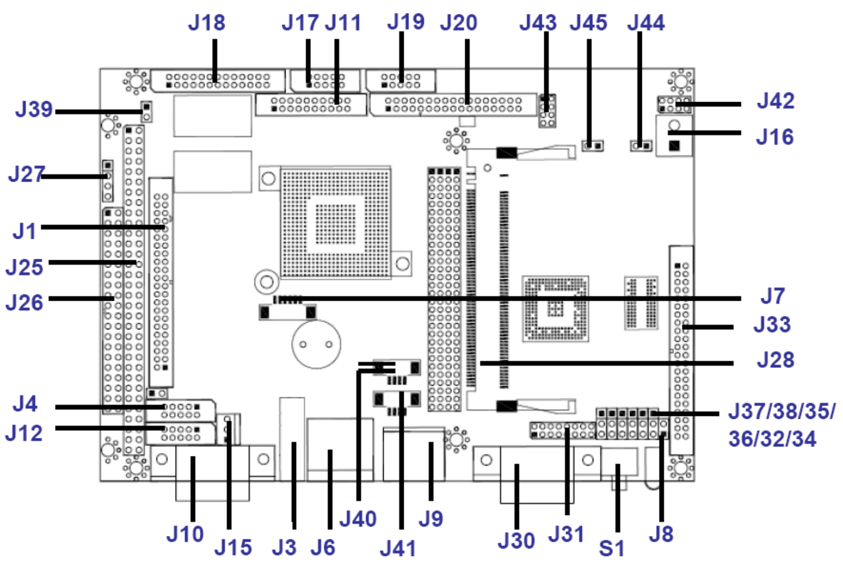
Power Requirement	+5V @ 800mA
Operating Temp.	-20 to +70 ℃
Operating Humidity	0% - 90% relative humidity, non-condensing
Size (W x H x D)	102 x 144 mm (4.01 x 5.67 inches)
Weight	150g

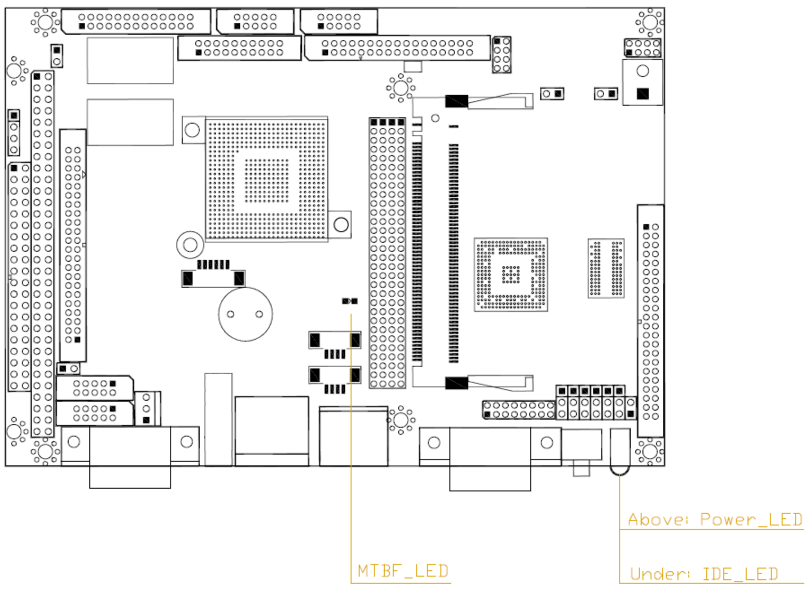
Note: The specification is subject to change without prior notice.

The Vortex86DX-SPARK Windows Embedded CE 6.0 Jump Start kit includes the following:

- 512MB IDE flash storage (Bootable)
- 100~240VAC 50/60Hz to +5VDC @ 2A power adapter
- Null RS-232 serial modem cable
- Cross-over RJ45 Ethernet cable
- 3 x RS-232 cable (flat ribbon cable, header to DB-9)
- Printer cable (flat ribbon cable, header to DB-25)
- IDE cable (flat ribbon cable)
- FDD cable (flat ribbon cable)
- USB cable (header to USB connector)
- 2 x LAN cable (header to RJ-45 jack)
- GPIO cable
- 2 x Audio cables
- Y-cable for PS/2 keyboard and mouse
- CD with Windows Embedded CE 6.0 BSP, SDK & jump start guide

VDX-6326 SBC Connectors, Jumpers and LED





Connectors and Jumpers Description

Nbr	Description	Type of Connections	Pin nbrs.
J1	IDE	Box Header, 2.0∅ ,22x2	44-pin
J2	CF Card Master/Slave Select	Pin Header, 2.54∅, 2x1	2-pin
J3	USB 2	USB connector	4-pin
J4	USB 1	Box Header,2.0∅ , 5x2	10-pin
J6	10/100Base-T Ethernet LAN	RJ45 Connector	8-pin
J7	JTAG	Wafer, 1.25∅ , 6x1	6-pin
J8	Reset	Pin Header, 2,54∅,1x2	2-pin
J9	PS/2 Keyboard / Mouse	Mini-DIN Female	6-pin
J10	COM1	D-Sub Male	9-pin
J11	GPIO (Port 0 / 1 /PWMx16)	Box Header, 2.0∅ ,10x2	20-pin
J12	**COM2**(RS232/RS485/RS422)	Box Header, 2.54∅ 5x2	10-pin
J15	RS-485	Molex Header,2.54∅, 3x1	3-pin
J16	Power Connector	Terminal Block 5.0∅,2x1	2-pin
J17	COM3	Box Header, 2.0∅ 5x2	10-pin
J18	PRINT	Box Header, 2.0∅ , 13x2	26-pin
J19	COM4	Box Header, 2.0∅ 5x2	10-pin
J20	FDD	Pin Header, 2.0∅ ,17x2	24-pin
J25	PC104 Connector – 64 pin	Box Header, 2.54∅ 32x2	64-pin
J26	PC104 Connector – 40 pin	Box Header, 2.54∅ 20x2	40-pin
J27	4P Power Source (Interconnect to PC/104 – **J25**)	Pin Header, 2.54∅ , 4x1	4-pin
J28	MINI_PCI _Type- Ⅲ	Type- Ⅲ connector	124-pin
J29	PC/104 + (Optional)	Box Header, 2.0∅ , 30x4	120-pin
J30	VGA	D-Sub Female	15-pin
J31	LVDS	Pin Header, 2.0∅ 8x2	16-pin
J32 | **J38**	Display type Setup	Pin Header, 2.54∅ , 3x1	3-pin
J33	LCD	Box Header,2.0∅ ,22x2	44-pin
J39	JTAG Disable (Default setting)	Pin Header, 2,54∅,1x2	2-pin
J40	LINE-OUT	Wafer, 1.25∅ , 4x1	4-pin
J41	MIC-IN	Wafer, 1.25∅ , 4x1	4-pin
J44	LAN2 Enable/Disable	Pin Header, 2.54∅, 2x1	2-pin
J45	LAN3 Enable/Disable	Pin Header, 2.54∅, 2x1	2-pin
CF1	Compact Flash	Type I/II CF Connector	50-pin
PWR_LED	POWER Active LED (Red)		
IDE_LED	IDE Active LED (Green)		
MTBF-LED	MTBF-Out (Orange)	LED-SMD	
SP1	BUZZER		
S1	RESET SWITCH		

Connectors and Jumpers Description

J1: IDE (44 Pins)

Pin #	Signal Name	Pin #	Signal Name
1	IDERST	2	GND
3	IDED7	4	IDED8
5	IDED6	6	IDED9
7	IDED5	8	IDED10
9	IDED4	10	IDED11
11	IDED3	12	IDED12
13	IDED2	14	IDED13
15	IDED1	16	IDED14
17	IDED0	18	IDED15
19	GND	20	NC
21	IDEREQ	22	GND
23	IDEIOW	24	GND
25	IDEIOR	26	GND
27	ICHRDY	28	GND
29	IDEACK	30	GND
31	IDEINT	32	NC
33	IDESA1	34	IDECBLID
35	IDESA0	36	IDESA2
37	IDECS-0	38	IDECS1
39	IDELED	40	GND
41	VCC	42	VCC
43	GND	44	NC

J2: CF Card Master / Slave Select

Pin #	Signal Name
CLOSE	Master
OPEN	Slave

J4: USB 1

Pin #	Signal Name	Pin #	Signal Name
1	VCC	2	VCC
3	LUSBD0-	4	LUSBD1-
5	LUSBD0+	6	LUSBD1+
7	GND	8	GND
9	GGND	10	GGND

J6: LAN1

Pin #	Signal Name	Pin #	Signal Name
1	TD+	2	TD-
3	RO+	4	NC
5	NC	6	RO-
7	NC	8	NC

J7: JTAG

Pin #	Signal Name	Pin #	Signal Name
1	VCC	2	GND
3	TCK	4	TDO
5	TDI	6	TMS

J8: RESET

Pin #	Signal Name	Pin #	Signal Name
1	Reset	2	GND

J9: PS/2 KBD / Mouse

Pin #	Signal Name	Pin #	Signal Name
1	KBCLK	2	MSCLK
3	GND	4	KBDAT
5	MSDAT	6	VCC
7	GGND	8	GGND
9	GGND		

J10: COM 1 (Optional: TTL/ GPIO-P4 / PWMx8)

Pin #	Signal Name	Pin #	Signal Name
1	DCD1	2	RXD1
3	TXD1	4	DTR1
5	GND	6	DSR1
7	RTS1	8	CTS1
9	RI1	10	GND
11	GND		

J11: GPIO (Port 0 / Port 1/PWMx16)

Pin #	Signal Name	Pin #	Signal Name
1	GND	2	VCC
3	GP00	4	GP10
5	GP01	6	GP11
7	GP02	8	GP12
9	GP03	10	GP13
11	GP04	12	GP14
13	GP05	14	GP15
15	GP06	16	GP16
17	GP07	18	GP17
19	VCC	20	GND

J12: COM2 RS232 / RS422 / RS485 (Change setting by BIOS)

Pin #	Signal Name	Pin #	Signal Name
1	DCD2/ **422TX-** / RS485-	2	RXD2 / **422TX+** / RS485+
3	TXD2 / **422RX+**	4	DTR2 / **422RX-**
5	GND	6	DSR2
7	RTS2	8	CTS2
9	RI2	10	NC

J15: RS485 (Auto direction)

Pin #	Signal Name
1	RS485＋
2	RS485－
3	GND

J16: Power Connector (Terminal Block 5.0mm)

Pin #	Signal Name
1	+5V
2	GND

J17: COM3

Pin #	Signal Name	Pin #	Signal Name
1	DCD3	2	RXD3
3	TXD3	4	DTR3
5	GND	6	DSR3
7	RTS3	8	CTS3
9	RI3	10	NC

J18: PRINT

Pin #	Signal Name	Pin #	Signal Name
1	STB-	14	AFD-
2	PD0	15	ERR-
3	PD1	16	INIT-
4	PD2	17	SLIN-
5	PD3	18	GND
6	PD4	19	GND
7	PD5	20	GND
8	PD6	21	GND
9	PD7	22	GND
10	ACK-	23	GND
11	BUSY	24	GND
12	PE	25	GND
13	SLCT	26	NC

J19: COM4

Pin #	Signal Name	Pin #	Signal Name
1	DCD4	2	RXD4
3	TXD4	4	DTR4
5	GND	6	DSR4
7	RTS4	8	CTS4
9	RI4	10	NC

J20: FDD

Pin #	Signal Name	Pin #	Signal Name
1	GND	2	DENSEL
3	GND	4	NC
5	GND	6	NC
7	GND	8	INDEX\
9	GND	10	MTRO\
11	GND	12	DS1\
13	GND	14	DS0\
15	GND	16	MTR1\
17	GND	18	DIR\
19	GND	20	STEP\
21	GND	22	WD\
23	GND	24	WG\
25	GND	26	TR0\
27	GND	28	WP\
29	GND	30	RD\
31	GND	32	HDSEL\
33	GND	34	DSKCHG\

J25: PC104 Connector – 64pin

Pin #	Signal Name	Pin #	Signal Name
1	IOCHCHK *	2	GND
3	SD7	4	RESETDRV
5	SD6	6	VCC
7	SD5	8	IRQ9
9	SD4	10	-5V
11	SD3	12	DRQ2
13	SD2	14	-12V
15	SD1	16	OWS
17	SD0	18	+12V
19	IOCHRDY	20	GND
21	AEN	22	SMEMW *
23	SA19	24	SMEMR *
25	SA18	26	IOW *
27	SA17	28	IOR *
29	SA16	30	DACK3 *
31	SA15	32	DRQ3
33	SA14	34	DACK1 *
35	SA13	36	DRQ1
37	SA12	38	REFRESH *
39	SA11	40	SYSCLK
41	SA10	42	IRQ7
43	SA9	44	IRQ6
45	SA8	46	IRQ5
47	SA7	48	IRQ4
49	SA6	50	IRQ3
51	SA5	52	DACK2 *
53	SA4	54	TC
55	SA3	56	BALE
57	SA2	58	VCC
59	SA1	60	OSC
61	SA0	62	GND
63	GND	64	GND

J26: PC104 Connector – 40pin

Pin #	Signal Name	Pin #	Signal Name
1	GND	2	GND
3	MEMCS16 *	4	SBHE *
5	IOCS16 *	6	SA23
7	IRQ10	8	SA22
9	IRQ11	10	SA21
11	IRQ12	12	SA20
13	IRQ15	14	SA19
15	IRQ14	16	SA18
17	DACK0 *	18	SA17
19	DRQ0	20	MEMR *
21	DACK5 *	22	MEMW *
23	DRQ5	24	SD8
25	DACK6 *	26	SD9
27	DRQ6	28	SD10
29	DACK7 *	30	SD11
31	DRQ7	32	SD12
33	VCC	34	SD13
35	MASTER *	36	SD14
37	GND	38	SD15
39	GND	40	NC

J27: 4P Power Source (Interconnect to PC/104 – J25)

Pin #	Signal Name
1	-5V
2	-12V
3	+12V
4	GND

J29: PC/104 + (Optional)

VI/O Default setting: +5V

If you need to use VI/O as +3.3V, please refer to the VDX-6326 user manual.

Pin #	A	B	C	D
1	GND	NC	+5V	AD00
2	VI/O(+5V)	AD02	AD01	+5V
3	AD05	GND	AD04	AD03
4	C/BE0#	AD07	GND	AD06
5	GND	AD09	AD08	GND
6	AD11	VI/O(+5V)	AD10	GND
7	AD14	AD13	GND	AD12
8	+3.3V	C/BE1#	AD15	+3.3V
9	SERR#	GND	NC	PAR
10	GND	PERR#	+3.3V	NC
11	STOP#	+3.3V	LOCK#	GND
12	+3.3V	TRDY#	GND	DEVSEL#
13	FRAME#	GND	IRDY#	+3.3V
14	GND	AD16	+3.3V	C/BE2#
15	AD18	+3.3V	AD17	GND
16	AD21	AD20	GND	AD19
17	+3.3V	AD23	AD22	+3.3V
18	IDSEL0	GND	IDSEL1	IDSEL2
19	AD24	C/BE3#	VI/O(+5V)	IDSEL3
20	GND	AD26	AD25	GND
21	AD29	+5V	AD28	AD27
22	+5V	AD30	GND	AD31
23	REQ0#	GND	REQ1#	VI/O(+5V)
24	GND	REQ2#	+5V	GNT0#
25	GNT1#	VI/O(+5V)	GNT2#	GND
26	+5V	CLK0	GND	CLK1
27	CLK2	+5V	CLK3	GND
28	GND	INTD#	+5V	RST#
29	+12V	INTA#	INTB#	INTC#
30	-12V	NC	NC	GND

J30: VGA

Pin #	Signal Name	Pin #	Signal Name
1	R OUT	2	G OUT
3	B OUT	4	NC
5	GND	6	GND
7	GND	8	GND
9	VCC	10	GND
11	NC	12	DDCDAT
13	HSYNC	14	VSYNC
15	DDCCLK		

J31: LVDS

Pin #	Signal Name	Pin #	Signal Name
1	VCC3 (3.3V)	2	VCC3 (3.3V)
3	GND	4	GND
5	RxIN0+	6	RxIN0-
7	RxIN1-	8	GND
9	GND	10	RxIN1+
11	RxIN2+	12	RxIN2-
13	CKIN-	14	GND
15	GND	16	CKIN+

J32~J38: Display type setup (CRT /LCD)

Connector	Pin #	Signal Name
J32	1	VCC
	2	GPIOA
	3	GND
J34	1	VCC
	2	GPIOB
	3	GND
J35	1	VCC
	2	GPIOC
	3	GND
J36	1	VCC
	2	GPIOD
	3	GND
J37	1	VCC
	2	GPIOE
	3	GND
J38	1	VCC
	2	GPIOF
	3	GND

J33: LCD (DVO) Connector

Pin #	Signal Name	Pin #	Signal Name
1	+3.3V	2	+3.3V
3	LG2	4	LG3
5	LG4	6	LG5
7	NC	8	NC
9	LR0	10	LR1
11	LR2	12	LR3
13	LR4	14	LR5
15	GND	16	NC
17	NC	18	NC
19	NC	20	GND
21	NC	22	NC
23	LB0	24	LB1
25	LB2	26	LB3
27	LB4	28	LB5
29	NC	30	NC
31	LG0	32	LG1
33	GND	34	GND
35	NC	36	LCLK
37	NC	38	LDE
39	NC	40	LHSYNC
41	NC	42	LVSYNC
43	LBACKL	44	LVDDEN

Please refer to user manul for TFT Flat Panel Data Output

J39: JTAG Disable (Default setting: Pin 1 & Pin 2 short)

Pin #	Signal Name	Pin #	Signal Name
1	GND	2	JTAG Disable

J40: LINE OUT

Pin #	Signal Name
1	LOUTR
2	GND
3	GND
4	LOUTL

J41: MIC-IN

Pin #	Signal Name
1	MICVREF
2	GND
3	GND
4	MIC-IN

J42: LAN2

Pin #	Signal Name	Pin #	Signal Name
1	TX+1	2	TX-1
3	RX+1	4	LED01
5	LED01+	6	RX-1
7	LED11+	8	LED11

J43: LAN3

Pin #	Signal Name	Pin #	Signal Name
1	TX+2	2	TX-2
3	RX+2	4	LED02
5	LED02+	6	RX-2
7	LED12+	8	LED12

J44: LAN2 Enable/Disable

Pin #	Signal Name
CLOSE	LAN Off
OPEN	LAN ON

J45: LAN3 Enable/Disable

Pin #	Signal Name
CLOSE	LAN Off
OPEN	LAN ON

VDX-6326 SBC System BIOS

The VDX-6326 SBC uses AMI BIOS. To reconfigure the SBC's BIOS settings, go through the following steps to enter BIOS configuration:

1. Immediately after power on, depress the <Del> key repeatedly to enter the system BIOS configuration mode.
2. Use the arrow up/down, left/right and PageUp/Page/Dn keys to navigate between different BIOS configuration options.
3. Press <Esc> to move back to previous menu.
4. To save changes, navigate to the **Save Changes and Exit** option from the main BIOS setting menu to save setting changes and Exit. (You can also press F10 to save the setting and exit.)

> Unless you are certain about the BIOS settings' function, do not change system BIOS configuration.
>
> Improper BIOS configuration can cause the VDX-6326 SBC fails to boot or function as expected.

VDX-6326 SBC System Memory, I/O and IRQ Mapping

Memory Mapping

Address	Description
0000:0000-9000:FFFF	System RAM
A000:0000-A000:FFFF	EGA/VGA Video Memory
B000:0000-B000:7FFF	MDA RAM, Hercules graphics display RAM
B000:8000-B000:FFFF	CGA display RAM
C000:0000-C000:7FFF	EGA/VGA BIOS ROM
C000:8000-C000:FFFF	Boot ROM enable.
D000:0000-D700:FFFF	Expansion ROM space.
D800:0000-DB00-FFFF	SPI Flash Emulate Floppy A Enable.
DC00:0000-DF00:FFFF	Expansion ROM space.
E000:0000-E000:FFFF	USB Legacy SCSI ROM space.
F000:0000-F000:FFFF	Motherboard BIOS

I/O Mapping

I/O Address	Device
0000h – 000Fh	8237 DMA Controller #1
0010h – 0017h	COM 9 **
0018h – 001Fh	Not use
0020h – 0021h	8259 Master Interrupt Controller
0022h – 0023h	6117D configuration port **

0024h – 002Dh	Not use
002Eh – 002Fh	Forward to LPC Bus
0030h – 003Fh	Not use
0040h – 0043h	Timer Counter 8254 **
0044h – 0047h	Not use
0048h – 004Bh	PWM counter 8254 **
004Ch – 004Dh	Not use
004Eh – 004Fh	Forward to LPC Bus
0050h – 005Fh	Not use
0060h	Keyboard data port
0061h	Port B + NMI control port
0062h – 0063h	8051 download 4K address counter
0064h	Keyboard status port
0065h	WatchDog0 reload counter
0066h	8051 download 8-bit data port
0067h	WatchDog1 reload counter
0068h – 006Dh	WatchDog1 control register
006Eh – 006Fh	Not use
0070h – 0071h	CMOS RAM port
0072h – 0075h	MTBF counter **
0076h – 0077h	Not use
0078h – 007Ch	GPIO port 0, 1, 2, 3, 4 default setup **
007Dh – 007Fh	Not use
0080h – 008Fh	DMA Page Registers
0090h – 0091h	Not use
0092h	System control register
0093h – 0097h	Not use
0098h – 009Ch	GPIO direction control **
00A0h – 00A1h	PIC 8259-2
00A2h – 00BFh	Not use
00C0h – 00DFh	8237 DMA Controller #2
00E0h – 00FFh	Not use
0100h – 0101h	GPCS1 default setting address
0170h – 0177h	IDE1 (IRQ-15)
01F0h – 01F7h	IDE0 (IRQ-14)
0220h – 0227h	Serial Port 8 Forward to LPC Bus **
0228h – 022Fh	Serial Port 7 Forward to LPC Bus **

0238h – 023Fh	Serial Port 6 Forward to LPC Bus **
0278h – 027Fh	Parallel port (IRQ-7, DMA 0) **
02E8h – 02EFh	Serial Port 4 (IRQ-11) **
02F8h – 02FFh	Serial Port 2 (IRQ-3)
0338h – 033Fh	Serial Port 5 Forward to LPC Bus **
00376h	IDE1 ATAPI device control write only register
03E8h – 03EFh	Serial Port 3 (IRQ-10) **
03F0h – 03F7h	Floppy Disk (IRA 6, DMA 2)
03F6h	IDE0 ATAPI device control write only register
03F8h – 03FFh	Serial Port 1 (IRQ-4)
0480h – 048Fh	DMA High page register
0490h – 0499h	Instruction counter register
04D0h – 04D1h	8259 Edge/Level control register **
0CF8h – 0CFFh	PCI configuration port
D400h – D4FFh	On board LAN
FC00h – FC05h	SPI Flash BIOS control register **
FC08h – FC0DH	External SPI Bus control register (output pin configurable GPIO3[0-3]) **

** Not in use for VDX-6326 SBC

IRQ Mapping

IRQ#	Device
IRQ0	System Timer
IRQ1	Keyboard Controller
IRQ2	Cascade for IRQ8 - 15
IRQ3	Serial Port 2
IRQ4	Serial Port 1
IRQ5	USB
IRQ6	Floppy
IRQ7	Parallel Port
IRQ8	Real Time Clock
IRQ9	USB, RTL8100B LAN1, RTL8100B LAN2
IRQ10	Serial Port 3
IRQ11	Serial Port 4
IRQ12	Mouse
IRQ13	Math Coprocessor
IRQ14	Hard Disk Controller
IRQ15	USB

DMA Mapping	
DMA#	**Description**
DMA0	
DMA1	
DMA2	Floppy Disk Controller
DMA3	
DMA4	
DMA5	
DMA6	
DMA7	

Appendix E – SBC Startup Options

By changing the BIOS settings, the VDX-6326 SBC can be configured to boot from the following resources.

- Internal IDE storage
- CompactFlash
- USB Floppy
- USB storage
- USB CD/DVD-ROM drive
- Remote Network Boot using PXE

By default, SBC is set to boot from the IDE storage.

Here are the steps to configure the SBC to boot from bootable USB flash storage. The USB flash storage device must be inserted to the SBC's USB interface prior to power on.

- Configure an USB flash storage to boot to DOS.

  USB flash devices from different manufacture are built with different components. Not all USB flash can be configured as bootable device. Refer to the USB flash manufacture's technical document for information.

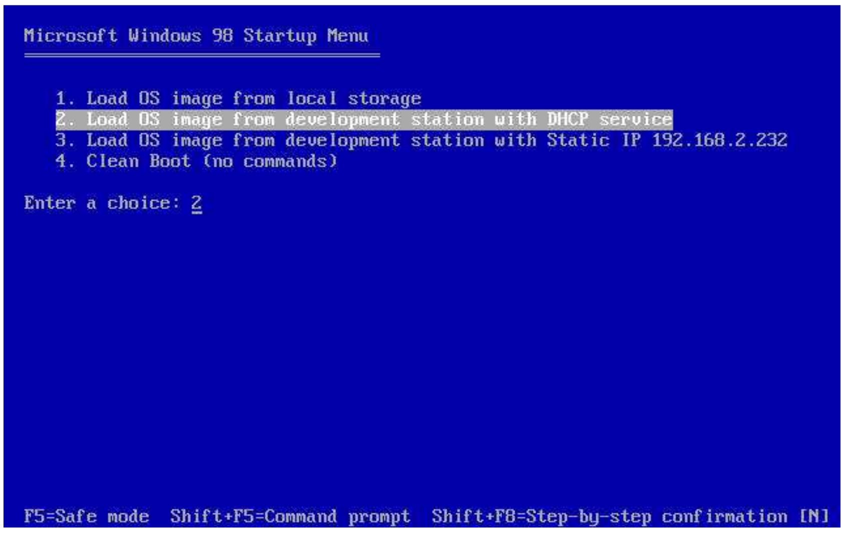
- Insert the bootable USB flash storage to one of the SBC's USB ports, before power on.
- Power on the SBC.
- Press the DEL key repeatedly immediately after SBC power on to enter BIOS configuration.
- From the BIOS Setup Utility menu, use the Left/Right Arrow key to select the "Boot" option.
- With the "Boot" option selected, use the Up/Down Arrow key to select "Boot Device Priority", and press the Enter key.
- Use the Up/Down Arrow key to select "1st Boot Device", and press the Enter key.
- From the small Options menu, use the Up/Down Arrow key to select "USB: xxx" (xxx = name of the USB flash storage) and press Enter.
- Press the ESC key to navigate back to the main menu.
- Use the Left/Right Arrow key to select "Exit"
- Use the Up/Down Arrow key to select "Save Changes and Exit" to complete the BIOS configuration.

Consult SBC manual for more information about BIOS settings.

When creating a bootable USB flash storage device, it's common to use DOS as the operating system. The DOS operating system supports FAT file system and can only address storage partition that is 2 GB or smaller.

Appendix F – Modify the VDX-6326 SBC's DOS Selection Menu

The VDX-6326 SBC has an internal IDE bootable flash storage preinstalled and configured to boot to DOS and provides multiple options to launch the Loadcepc.exe boot loader with different command line options as shown in figure F1.



```
Microsoft Windows 98 Startup Menu
═════════════════════════════════

   1. Load OS image from local storage
   2. Load OS image from development station with DHCP service
   3. Load OS image from development station with Static IP 192.168.2.232
   4. Clean Boot (no commands)

Enter a choice: 2

F5=Safe mode  Shift+F5=Command prompt  Shift+F8=Step-by-step confirmation [N]
```

Fig. F1 - VDX-6326 SBC startup menu (with CE 6.0 jump start kit software preinstalled)

The default selection, will execute after 15 seconds delay. During the initial 15 seconds delay, you can select and execute the other options, using the arrow up/down key.

You can change the default selected option and the delay time. To change the default option, use a text editor to edit the CONFIG.SYS file located in the root of the SBC's storage as follow.

Let's say you are using the Cross-over Ethernet cable to connect your development workstation directly to the SBC. You need to configure your development workstation and SBC to use static IP address. It would be convenience for the SBC to launch "Option 3", "Load OS image from development station with Static IP 192.168.2.232", as the default selection. To save time, it's also good to shorten the delay time from 15 seconds to 5 seconds. Here is how you can make the changes.

Use a text editor to edit the CONFIG.SYS file on the SBC as follow:

Locate the following entry within CONFIG.SYS:

Menudefault=1LOCAL, 15

Change the the above entry to the following:

Menudefault=3STATIC, 5

If you are not familiar with DOS batch file, take a look at the AUTOEXEC.BAT file locates at the root of the file system. When the SBC is power up, it executes this batch file first and uses the configuration settings in the CONFIG.SYS file.

The DOS operating system and the SBC's startup selection menu are designed to help provide an efficient development environment. When deploying the product to distribution using the Windows Embedded CE operating system, use BIOSLoader in place of DOS to launch the OS runtime image.

BIOSLoader is a boot loader, included with Windows Embedded CE 6.0, in the following folder:

\WINCE600\Platform\CEPC\SRC\Bootloader\BIOSLoader\

Source code, binary and setup floppy image are provided. Refer to the online documentation for more information.

Here is the source code listing for the Config.sys file, the DOS startup configuration file for the VDX-6326 SBC:

```
[menu]
menuitem=1LOCAL, Load OS image from local storage
menuitem=2DHCP, Load OS image from development station with DHCP service
menuitem=3STATIC, Load OS image from development station with Static IP 192.168.2.232
menuitem=4CLEAN, Clean Boot (no commands)
menudefault=1LOCAL,15
menucolor=7,1

[1LOCAL]

[2DHCP]

[3STATIC]

[4CLEAN]

[COMMON]
buffers=10,0
files=30
break=on
lastdrive=Z
dos=high,umb
device=himem.sys /testmem:OFF
```

Here is the source code listing for the Autoexec.bat file, the DOS startup batch file for the VDX-6326 SBC:

```
@echo off
verify off
PROMPT $p$g

set NET_IRQ=5
set NET_IOBASE=EC00
set NET_IP=

if "%CONFIG%" == "1LOCAL"  goto 1LOCAL
if "%CONFIG%" == "2DHCP"   goto 2DHCP
if "%CONFIG%" == "3STATIC" goto 3STATIC
if "%CONFIG%" == "4CLEAN"  goto 4CLEAN

:1LOCAL
Loadcepc nk.bin
goto END

:2DHCP
goto REMOTE

:3STATIC
Set NET_IP=192.168.2.232
goto REMOTE

:REMOTE
loadcepc /C:1 /e:%NET_IOBASE%:%NET_IRQ%:%NET_IP% eboot.bin
goto END

:4CLEAN

:END
```

Appendix G – Debug Serial Port

Windows Embedded CE captures the first Serial port and uses it to output debug messages. Since much of the debug messages are available with the use of KITL via an Ethernet connection.

When using one of the preconfigured options to load the CE OS runtime image from the local storage, the Loadcepc is executed with command line parameter as follow:

```
Loadcepc NK.bin
```

The NK.bin file is the Windows Embedded CE OS runtime image. To configure COM1 to send out serial debug messages, change the above line of code to the following:

```
Loadcepc /C:1 NK.bin
```

When using one of the preconfigured options to send bootme request and download the runtime image from the development workstation, the Loadcepc is executed with command line parameter as follow:

```
Loadcepc /C:1 /e:%NET_IOBASE%:%NET_IRQ%:%NET_IP% EBOOT.bin
```

The EBOOT.bin file is an Ethernet boot loader, needed to request and process the function to download OS runtime images from the Platform Builder development workstation.

To send request to download runtime image from the development workstation using static IP (192.168.2.232) and configure the device to send serial debug messages via COM2, launch the Loadcepc executable with command parameters as follow:

```
Loadcepc /C:2 /e:0:0:192.168.2.232 EBOOT.bin
```

To access help information for the Loadcepc loader, launch the Loadcepc executable with the command parameters as follow:

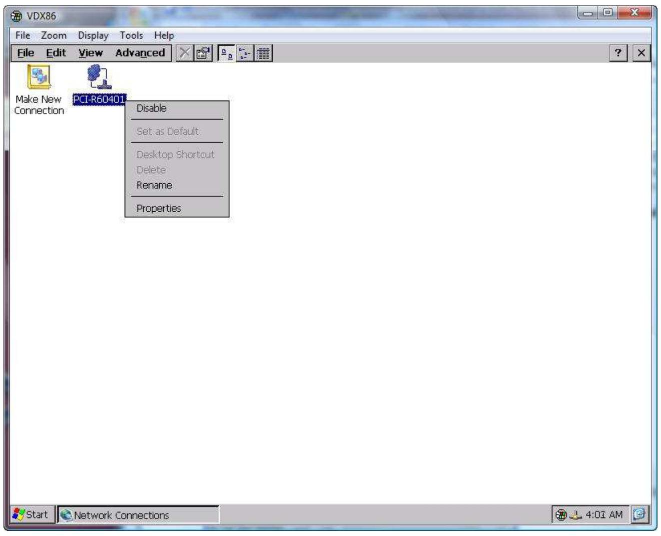
```
Loadcepc /?
```

Appendix H – Using Static IP Address

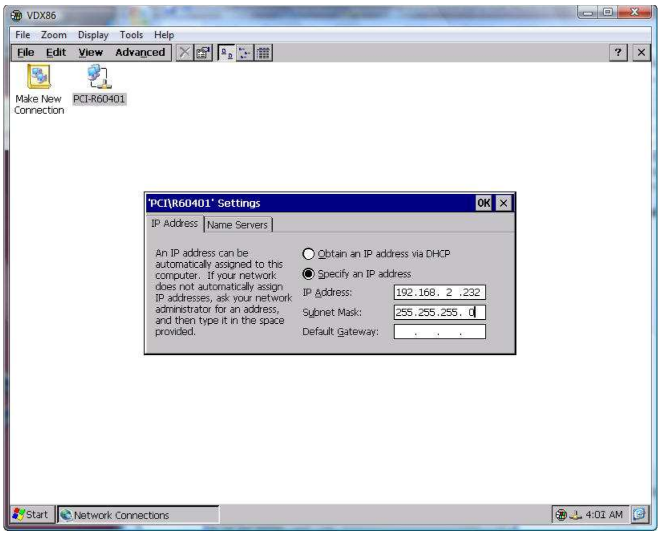
To use Static IP address, the IP address for the SBC and development workstation must be set to the same subnet.

The DOS boot-loader pre-installed on the SBC is configured with the Static IP address, 192.168.2.232, when option 5 is selected (Load OS image from development workstation with Static IP). This IP address is use for the **Loadcepc** bootloader only and is not being passed to the CE 6.0 runtime image launch by the **Loadcepc** bootloader. After the CE 6.0 runtime image is successfully download from the development workstation and launched, the IP address for SBC must be configured manually.

From Windows Embedded CE desktop, select Start **| Settings | Network and Dial-up Connections** to bring up the **Network Connections** screen.



From the Network Connection screen, right click on **PCI-R60401** and select Properties.



'PCI\R60401' Settings

IP Address | Name Servers

An IP address can be automatically assigned to this computer. If your network does not automatically assign IP addresses, ask your network administrator for an address, and then type it in the space provided.

○ Obtain an IP address via DHCP
● Specify an IP address

IP Address: 192.168. 2 .232
Subnet Mask: 255.255.255. 0
Default Gateway: . . .

From the PCI\R60401 Settings screen, enter the static IP address.

Note: You can use a different IP address.
You need to configure your development workstation's IP address to the same subnet as SBC.

If your development environment requires the use of static IP address, it's more convenience to build a CE 6.0 OS image with a static IP address preconfigured.

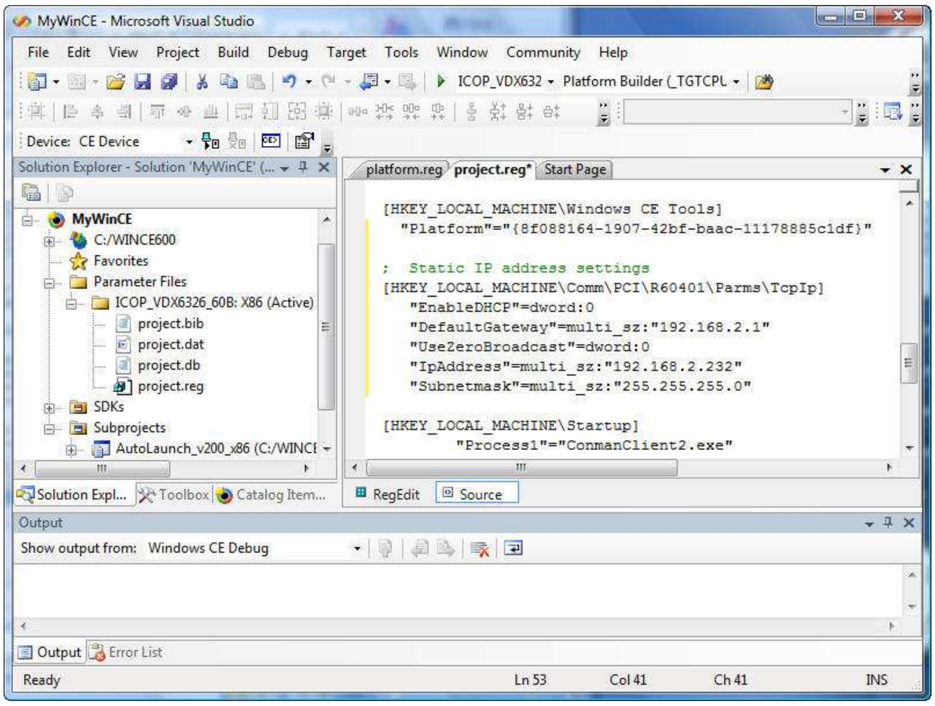
You can configure the OS design to generate an OS runtime image with a preconfigured static IP address by adding the following registry entries to the OS design project's PROJECT.REG registry file.

```
[HKEY_LOCAL_MACHINE\Comm\PCI\R60401\Parms\TcpIp]
  "EnableDHCP"=dword:0
  "DefaultGateway"=multi_sz:"192.168.2.1"
  "UseZeroBroadcast"=dword:0
  "IpAddress"=multi_sz:"192.168.2.232"
  "Subnetmask"=multi_sz:"255.255.255.0"
```

To add the above registry entries to the OS design, with the OS design project open, perform the following steps.

- Click and select **Solution Explorer** from VS2005 IDE
- Expand the **Parameter Files** folder
- Expand **the ICOP_VDX6326_60B: X86** folder
- Double click on the **project.reg** file
- In the center pane, click on the Source tap to view in source format
- Add the above registry entries to the end of the file.

Your VS2005 should look similar to the following.



```
[HKEY_LOCAL_MACHINE\Windows CE Tools]
  "Platform"="{8f088164-1907-42bf-baac-11178885c1df}"

;  Static IP address settings
[HKEY_LOCAL_MACHINE\Comm\PCI\R60401\Parms\TcpIp]
   "EnableDHCP"=dword:0
   "DefaultGateway"=multi_sz:"192.168.2.1"
   "UseZeroBroadcast"=dword:0
   "IpAddress"=multi_sz:"192.168.2.232"
   "Subnetmask"=multi_sz:"255.255.255.0"

[HKEY_LOCAL_MACHINE\Startup]
        "Process1"="ConmanClient2.exe"
```

MyWinCE - Microsoft Visual Studio
File Edit View Project Build Debug Target Tools Window Community Help
ICOP_VDX632
Platform Builder (_TGTCPL
Device: CE Device
Solution Explorer - Solution 'MyWinCE' (...
MyWinCE
C:/WINCE600
Favorites
Parameter Files
ICOP_VDX6326_60B: X86 (Active)
project.bib
project.dat
project.db
project.reg
SDKs
Subprojects
AutoLaunch_v200_x86 (C:/WINCE
Solution Expl...
Toolbox
Catalog Item...
platform.reg
project.reg*
Start Page
RegEdit
Source
Output
Show output from: Windows CE Debug
Output
Error List
Ready
Ln 53
Col 41
Ch 41
INS

After rebuilding the OS design, the resulting CE 6.0 image will have the pre-assigned static IP address.

Appendix I – *Recover Jump Start Kit's Original Files*

The VDX-6326 SBC is preconfigured to boot to Microsoft DOS 6.22 and the necessary files to load the prebuilt CE 6.0 OS runtime images and Ethernet boot loader to download runtime image from the development workstation.

VDX-6326 SBC Preconfigured Files

The following files are preconfigured in the root of the SBC's FAT16 file system:

- Loadcepc.exe
  This is a DOS boot loader for Windows Embedded CE. It can be used to launch a CE 6.0 OS runtime image from the local storage. It can also be used to launch the Ethernet boot loader, Eboot.bin, to establish connectivity with the Platform Builder development workstation and download runtime image from the development workstation to the target device.

- Nk.bin
  This is the prebuilt CE 6.0 OS runtime image for the jump start kit.

- Vesatest.exe
  This utility is used to check the system's supported VESA video display mode.

- Eboot.bin
  This is the Ethernet boot loader. It's launched by Loadcepc.exe to establish connectivity with the Platform Builder development workstation and download Windows Embedded CE OS runtime image from the development workstation to the target device.

- Himem.sys
  This is a DOS device driver to enable DOS to use memory beyond the original 640K limitation, also refer to as extended memory region.

  Note: There are multiple version of DOS, when using DOS 6.22 and Windows 98 DOS, Loadcepe.exe requires Himem.sys to function.

  When using Windows Millennium DOS, Himem.sys is not required.

- Autoexec.bat
  This is an auto execution batch file executed by DOS when the system starts.
  The file is in ASCII text format and can be viewed or edited using any text editor.
  The Edit.com, a simple text editor, is provided to edit this file.

- Config.sys
  This is the system startup configuration file.
  The file is in ASCII text format and can be viewed or edited using any text editor.
  The Edit.com, a simple text editor, is provided to edit this file

The VDX-6326 SBC preconfigured files are provided on the CD-ROM that comes with the Vortex86DX-SPARK Windows Embedded CE 6.0 jump start kit, in the following directories:

- \MSJK_Files

Recover Damaged Files

To recover individual file or damaged file system, you can perform one of the following:

- Copy the file(s) from the jump start CD-ROM to the SBC's local storage, overwritten the existing file(s).
- Format the SBC's IDE flash file system and recover all of the original files from the CD-ROM.

Note: Unless the SBC's flash storage file system is corrupted and not able to boot, it's best to copy the needed files from the jump start CD-ROM and avoid formatting the flash.

To perform the above recovery tasks, you need to boot SBC from an USB floppy, USB flash storage or USB CD-ROM to recover the file(s).

The VDX-6326 SBC can be configured to boot from the following sources:

- IDE storage
- USB floppy
- USB CD-ROM
- USB flash storage
- LAN

One of the convenience methods to recover damaged file is to use a bootable USB flash storage, as the file transfer medium, to copy file(s) from the jump start CD-ROM to the SBC's flash storage.

Preparing USB Flash Storage to Boot to DOS

Before configuring the SBC to boot from USB flash storage, you need to prepare the USB flash storage to be able to boo to DOS.

Keep in mind there are many different type of USB flash manufactured by different companies with different hardware, and not all of them can be configured to boot to DOS.

It's important to know that DOS can only address FAT file partition that is 2GB or smaller. So, you need to use an USB flash storage that is 2GB or smaller in size.

For some USB flash storage, making it bootable to DOS can be done with the following simple steps:

- Format the USB flash storage using FAT or FAT16 (not FAT32).
- Copy the following DOS system file to the USB flash storage.
  - IO.SYS
  - MSDOS.SYS
  - COMMAND.COM

There are different version of DOS, such as 6.22, Windows 98 and Windows Millennium. You need to use DOS files from the same version.

You can find these file by searching on the Web using the "DOS boot disk" key word.

Some of the USB flash storage manufacture provide utility to format and configure the USB flash to boot to DOS.

Due to the large pool of available USB flash storages; there is not a procedure or method to make the USB flash bootable to DOS that works for all USB flash in the market.

Some of the USB flash storage devices are built with file encryption, support multiple function and etc. Usually, these multi-function USB flash storage devices cannot be configured to boot to DOS. It's best to use a simple USB flash storage device that is 2GB or smaller.

In addition to formatting and configuring the USB flash storage to boot to DOS, you also need to copy the necessary DOS utilities to the USB flash storage and use them to format the SBC's IDE flash storage.

If the USB flash storage is made bootable to Windows Millennium or Windows 98, copy the following DOS utilities to the USB flash storage (make sure to copy the correct version of utility):

- Fdisk.exe
- Format.com
- Attrib.exe

If the USB flash storage is made bootable to DOS 6.22, in addition to the above utilities, the following are needed:

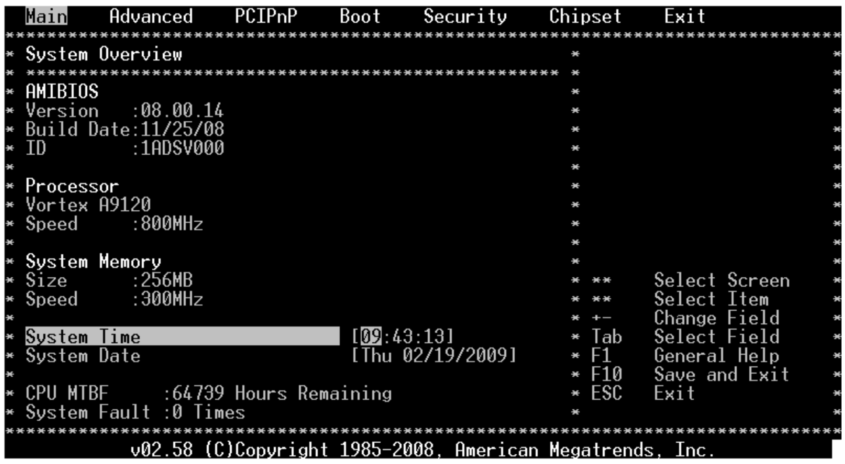
- Sys.com (Needed to transfer the DOS system file to SBC's IDE flash storage)
- Himem.sys (Extended memory driver needed for DOS 6.22 to support Loadcepc)

Configure the VDX-6326 SBC to Boot from USB Flash Storage

Assuming you have a bootable USB flash storage, insert the USB flash storage to the SBC and turn on the power. The USB flash storage must be inserted to the SBC's USB interface prior to power on, in order for the system BIOS to detect its present.

Improper BIOS settings affect system behavior and prevent the system from booting and function as expected.

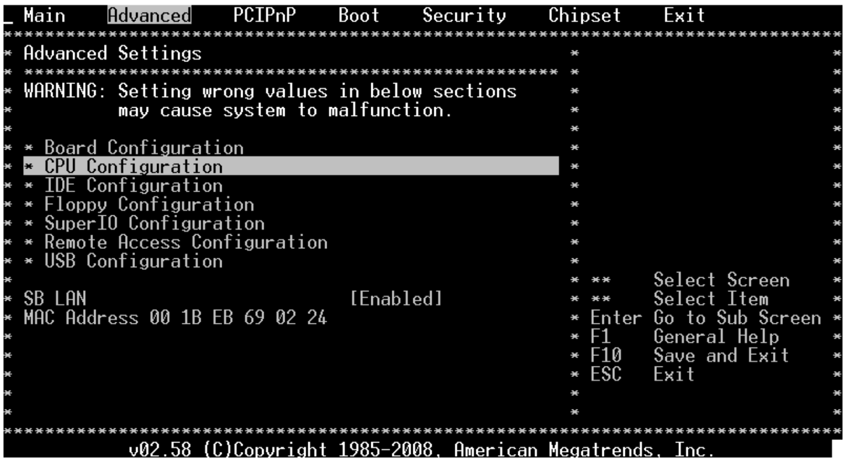
Immediately after power on, press the "Del" key repeatedly to bring up the BIOS configuration menu, similar to the screen in Figure I-1.



```
  Main    Advanced    PCIPnP    Boot    Security    Chipset    Exit
* System Overview                                  *
* AMIBIOS                                          *
* Version    :08.00.14                             *
* Build Date:11/25/08                              *
* ID         :1ADSV000                             *
*                                                  *
* Processor                                        *
* Vortex A9120                                     *
* Speed      :800MHz                               *
*                                                  *
* System Memory                                    *
* Size       :256MB                                * **    Select Screen
* Speed      :300MHz                               * **    Select Item
*                                                  * +-    Change Field
* System Time                 [09:43:13]           * Tab   Select Field
* System Date                 [Thu 02/19/2009]     * F1    General Help
*                                                  * F10   Save and Exit
* CPU MTBF      :64739 Hours Remaining             * ESC   Exit
* System Fault :0 Times                            *
           v02.58 (C)Copyright 1985-2008, American Megatrends, Inc.
```

Figure I-1

From the main BIOS configuration menu, use the left/right arrow key to navigate to the Advanced settings menu, similar to Figure I-1.



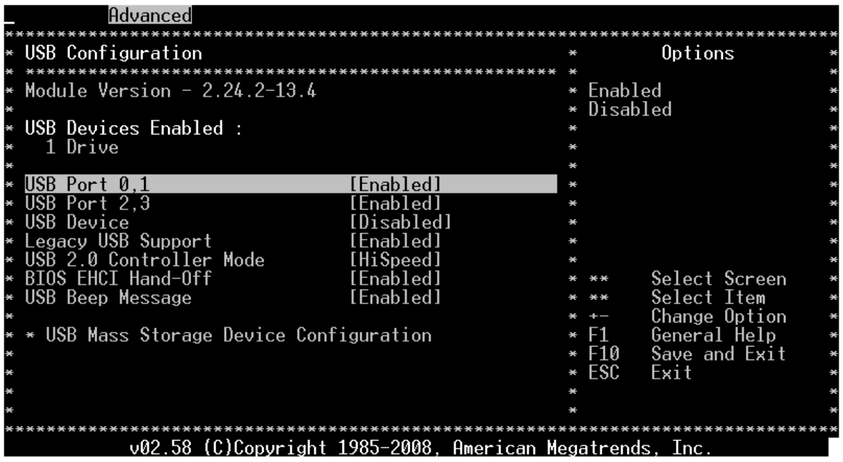
```
_ Main    Advanced    PCIPnP    Boot    Security    Chipset    Exit
********************************************************************************
* Advanced Settings                                    *                        *
* **************************************************** *                        *
* WARNING: Setting wrong values in below sections      *                        *
*          may cause system to malfunction.            *                        *
*                                                      *                        *
* * Board Configuration                                *                        *
* * CPU Configuration                                  *                        *
* * IDE Configuration                                  *                        *
* * Floppy Configuration                               *                        *
* * SuperIO Configuration                              *                        *
* * Remote Access Configuration                        *                        *
* * USB Configuration                                  *                        *
*                                                      * **     Select Screen   *
* SB LAN                         [Enabled]             * **     Select Item     *
* MAC Address 00 1B EB 69 02 24                        * Enter Go to Sub Screen *
*                                                      * F1     General Help    *
*                                                      * F10    Save and Exit   *
*                                                      * ESC    Exit            *
*                                                      *                        *
*                                                      *                        *
********************************************************************************
            v02.58 (C)Copyright 1985-2008, American Megatrends, Inc.
```

Figure I-2

The Advanced BIOS settings screen may not show the following hidden options:

- Board Configuration
- Floppy Configuration
- SuperIO Configuration

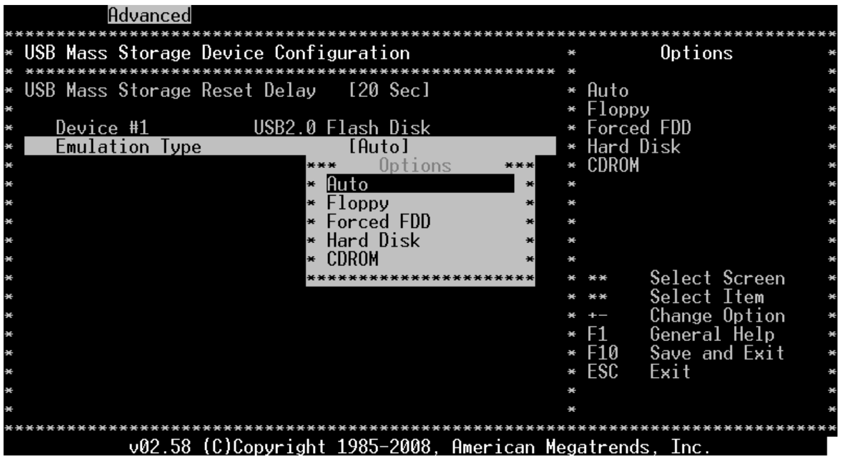
From the Advanced BIOS settings screen, use the Up/Down arrow key to navigate to the “USB configuration” option and press Enter to bring up the USB configuration screen, as shown in Figure I-3.



```
_         Advanced
********************************************************************************
* USB Configuration                                    *        Options         *
* **************************************************** *                        *
* Module Version - 2.24.2-13.4                         * Enabled                *
*                                                      * Disabled               *
* USB Devices Enabled :                                *                        *
*   1 Drive                                            *                        *
*                                                      *                        *
* USB Port 0,1                   [Enabled]             *                        *
* USB Port 2,3                   [Enabled]             *                        *
* USB Device                     [Disabled]            *                        *
* Legacy USB Support             [Enabled]             *                        *
* USB 2.0 Controller Mode        [HiSpeed]             *                        *
* BIOS EHCI Hand-Off             [Enabled]             * **     Select Screen   *
* USB Beep Message               [Enabled]             * **     Select Item     *
*                                                      * +-     Change Option   *
* * USB Mass Storage Device Configuration              * F1     General Help    *
*                                                      * F10    Save and Exit   *
*                                                      * ESC    Exit            *
*                                                      *                        *
*                                                      *                        *
********************************************************************************
            v02.58 (C)Copyright 1985-2008, American Megatrends, Inc.
```

Figure I-3

From the USB configuration screen, use the Up/Down arrow key to navigate to the “USB Mass Storage Device Configuration” and press enter to bring up the USB Mass Storage Device Configuration screen, as shown in Figure I-4.



```
          Advanced
*************************************************************************
* USB Mass Storage Device Configuration            *        Options     *
* ************************************************ *                    *
* USB Mass Storage Reset Delay   [20 Sec]          * Auto               *
*                                                  * Floppy             *
*    Device #1          USB2.0 Flash Disk          * Forced FDD         *
*    Emulation Type              [Auto]            * Hard Disk          *
*                            ***    Options    *** * CDROM              *
*                            * Auto              * *                    *
*                            * Floppy            * *                    *
*                            * Forced FDD        * *                    *
*                            * Hard Disk         * *                    *
*                            * CDROM             * *                    *
*                            ********************* * **     Select Screen *
*                                                  * **     Select Item   *
*                                                  * +-     Change Option *
*                                                  * F1     General Help  *
*                                                  * F10    Save and Exit *
*                                                  * ESC    Exit          *
*                                                  *                    *
*************************************************************************
          v02.58 (C)Copyright 1985-2008, American Megatrends, Inc.
```

Figure I-4

The "Device #1" setting list the detected USB flash storage device. The device name listed will be different for each different type of USB flash storage from different company.

The default Emulation Type is set to "Auto", which works for some device and does not work for some device.

> You may have to go through multiple tries to find a setting that work.

For most USB flash storage device, setting the Emulation Type to "Hard Disk" yield the best result.

> After the file recovery is done, be sure to change the Emulation setting back to Auto. Otherwise, USB device may not function properly under CE 6.0.

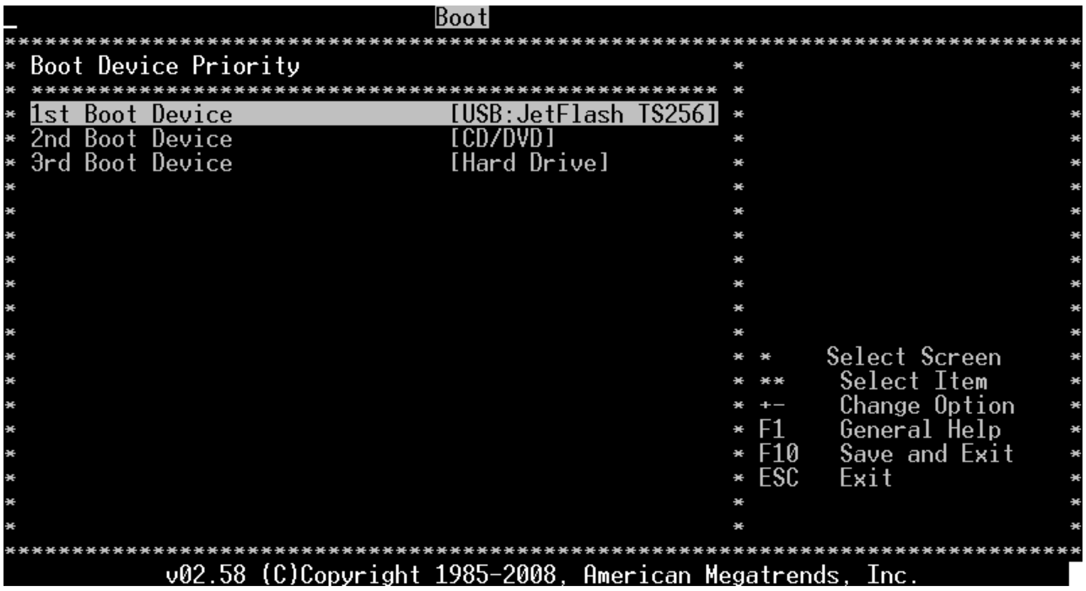
Boot Device Priority

In addition to configuring the USB flash storage device emulation type, the device boot sequence also affect the SBC's boot process.

Press the "Esc" key to go back to the previous menu. You need to press the "Esc" a few times to go back to the main BIOS settings menu.

From the main BIOS setting menu, use the Left/Right arrow key to move to the Boot setting menu.

From the Boot Settings menu, use the Up/Down arrow key to navigate to the Boot Device Priority option and press enter to bring up the Boot Device Priority screen, as shown in Figure I-5.



```
                              Boot
****************************************************************************
* Boot Device Priority                                *                    *
* ************************************************** *                    *
* 1st Boot Device              [USB:JetFlash TS256]   *                    *
* 2nd Boot Device              [CD/DVD]               *                    *
* 3rd Boot Device              [Hard Drive]           *                    *
*                                                     * *    Select Screen *
*                                                     * **    Select Item  *
*                                                     * +-    Change Option*
*                                                     * F1    General Help *
*                                                     * F10   Save and Exit*
*                                                     * ESC   Exit         *
****************************************************************************
          v02.58 (C)Copyright 1985-2008, American Megatrends, Inc.
```

Figure I-5

There may be two devices shown on the screen. Select the 1st Boot Device and press enter to bring up the Options screen, and select the USB flash storage device as the 1st Boot Device and press enter.

Press the "F10" key to save the settings and exit BIOS configuration.

The system will reset and boot from the USB flash storage.

After file recovery is done, reconfigure the BIOS to boot from the IDE device.

Steps to Recover All Files (USB Flash boot to DOS 6.22)

Assuming the SBC and bootable USB flash storage configuration were successful, with the SBC able to boot DOS from USB flash storage, work through the following steps to recover the VDX-6326 SBC files:

- Copy the following files from the VDX-6326 SBC jump start CD to the USB flash storage:
  - \Recover\Autoexec.bat
  - \Recover\Config.sys
  - \Recover\Eboot.bin
  - \Recover\Loadcepc.exe
  - \Recover\Himem.sys
  - \Recover\Nk.bin

- Boot SBC from the USB flash storage.
- Format the SBC's IDE flash storage using the following command:
  - Format d: /s

Important: If the SBC is still able to boot to DOS from the IDE flash storage, it's not needed to format the flash. Instead, just copy the files from the

Jump Start CD-ROM to the SBC’s IDE flash storage and overwritten the corrupted file.

The USB flash storage is emulated as hard drive and become drive C: after booting to DOS, with drive D: assigned to the SBC’s IDE flash storage.

If the USB flash storage is emulated as the floppy drive and become drive A: after booting to DOS, the drive letter C: will be assigned to the SBC’s IDE flash storage. In this case, use the following format command:

```
Format C: /s
```

The “/s” parameter instruct the Format utility to transfer DOS system to the IDE flash after formatting.

- Copy the following files from the USB flash storage to the SBC’s IDE flash storage:
  - Autoexec.bat
  - Config.sys
  - Eboot.bin
  - Himem.sys
  - Loadcepc.exe
  - Nk.bin

Remove the USB flash storage and reboot SBC. Immediately after SBC power reset, press the “Del” key to enter BIOS configuration and navigate to the Boot Settings menu to configure the Boot Device Priority to configure the SBC’s IDE flash storage as the 1st boot device.

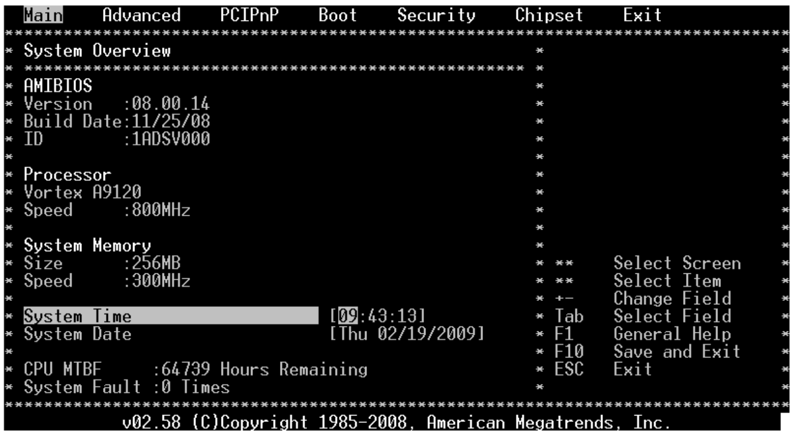
Note: If the IDE flash storage is recovered with a different version of DOS, the Config.sys, the DOS startup configuration file, may need to be modified to meet the particular version of DOS in use.

Appendix J – VDX-6326 SBC's System BIOS

The VDX-6326 SBC uses AMI BIOS.

To access the SBC's BIOS configuration, immediately after power on, or power reset, press the "Del" key repeatedly.

After successful gaining access to the system's BIOS configuration, the main BIOS configuration screen will show, similar to Figure J-1.



```
 Main    Advanced    PCIPnP    Boot    Security    Chipset    Exit
* System Overview                                   *
* ************************************************* *
* AMIBIOS                                           *
* Version    :08.00.14                              *
* Build Date:11/25/08                               *
* ID         :1ADSV000                              *
*                                                   *
* Processor                                         *
* Vortex A9120                                      *
* Speed      :800MHz                                *
*                                                   *
* System Memory                                     *
* Size       :256MB                                 * **    Select Screen
* Speed      :300MHz                                * **    Select Item
*                                                   * +-    Change Field
* System Time                     [09:43:13]        * Tab   Select Field
* System Date                     [Thu 02/19/2009]  * F1    General Help
*                                                   * F10   Save and Exit
* CPU MTBF      :64739 Hours Remaining              * ESC   Exit
* System Fault :0 Times                             *
            v02.58 (C)Copyright 1985-2008, American Megatrends, Inc.
```

Figure J-1

************************   IMPORTANT   ************************

Incorrect BIOS settings may cause the system fail to function as expected,

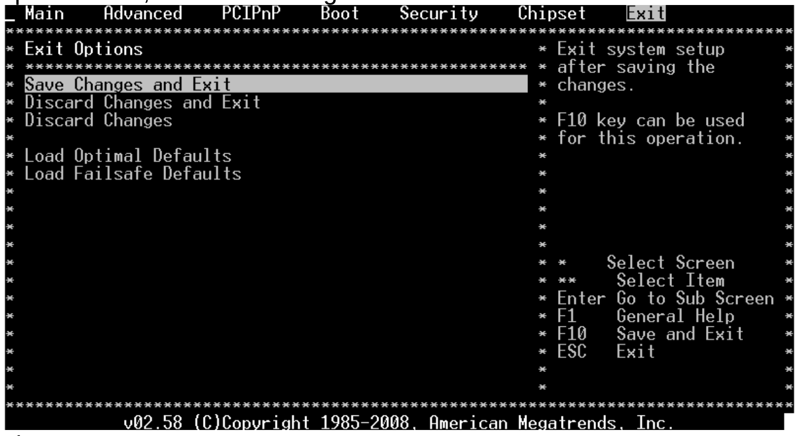
And may prevent the system from completing the boot process.

****

Default BIOS Settings

When the system fails to function due to incorrect BIOS settings, go through the following steps to configure the BIOS with the default settings:

- Enter BIOS configuration mode (Press "Del" key repeatedly immediately after powering on the SBC)
- From the main BIOS configuration menu, use the Left/Right arrow key to navigate to the Exit option menu, as shown in Figure J-2.



```
_ Main    Advanced    PCIPnP    Boot    Security    Chipset    Exit
* Exit Options                                       * Exit system setup
* ************************************************** * after saving the
* Save Changes and Exit                              * changes.
* Discard Changes and Exit                           *
* Discard Changes                                    * F10 key can be used
*                                                    * for this operation.
* Load Optimal Defaults                              *
* Load Failsafe Defaults                             *
*                                                    * *     Select Screen
*                                                    * **    Select Item
*                                                    * Enter Go to Sub Screen
*                                                    * F1    General Help
*                                                    * F10   Save and Exit
*                                                    * ESC   Exit
            v02.58 (C)Copyright 1985-2008, American Megatrends, Inc.
```

Figure J-2

- From the Exit Options menu, use the Up/Down arrow key to navigate to the "Load FailSafe Defaults" and press the "Enter" key.
- When the "Load Failsafe Defaults?" screen comes up, select OK.
- From the Exit Options menu, use the Up/Down arrow key to navigate to the "Load Optimal Defaults" and press the "Enter" key.
- Press the "Esc" key to go back to the main BIOS setting menu.
- From the main BIOS setting menu, use the Left/Right arrow key to navigate to the Advanced Settings menu.
- Use the Up/Down arrow key to navigate to the IDE Configuration option and press enter to bring up the IDE Configuration menu.
- From the IDE Configuration menu, use the Up/Down arrow key to navigate to the "OnBoard IDE Operate Mode" and press "Enter" to bring up the Options dialog box.
- Select "Native Mode and press "Enter"
- Press the "Esc" key to go back to the main BIOS setting menu.
- Use the Right/Left arrow key to navigate to the Advanced PCI/PnP settings screen.
- Use the Up/Down arrow key to navigate to the PCI IDE BusMaster option and press "Enter".
- From the Options dialog box, select "Enabled" and press "Enter".

BIOS Settings Impacting IDE Storage in CE 6.0

The ICOP_VDX6326_60B BSP is configured to support the VDX-6326 SBC preconfigured with the following IDE and PCI-PnP BIOS settings:

- The "**OnBoard IDE Operate Mode**" setting, one of the Advanced BIOS settings, is configured to "**Native Mode**".

- The "**PCI IDE BusMaster**" setting, one of the Advanced PCI/PnP settings, is configured to "**Enabled**".

In addition to the above, check to make sure the 1st Boot Device, from the Boot Device Priority settings, is configured to boot from the correct device.

Appendix K – Configure System BIOS to Boot from USB Flash Storage

The VDX-6326 SBC uses AMI BIOS.

To access the SBC's BIOS configuration, immediately after power on, or power reset, press the "Del" key repeatedly.

*********************************** IMPORTANT **********************************

Incorrect BIOS settings may cause the system fail to function as expected,

And may prevent the system from completing the boot process.

****

Preparing USB Flash Storage to Boot to DOS

Before configuring the SBC to boot from USB flash storage, you need to prepare the USB flash storage to be able to boo to DOS.

Keep in mind there are many different type of USB flash manufactured by different companies with different hardware, and not all USB flash storage device can be configured to boot to DOS.

It's important to know that DOS can only address FAT file partition that is 2GB or smaller. So, you need to use an USB flash storage that is 2GB or smaller in size.

For some USB flash storage, making it bootable to DOS can be done with the following simple steps:

- Format the USB flash storage using FAT or FAT16 (not FAT32).
- Copy the following DOS system file to the USB flash storage.
  - IO.SYS
  - MSDOS.SYS
  - COMMAND.COM

There are different version of DOS, such as 6.22, Windows 98 and Windows Millennium. You need to use DOS files from the same version.

You can find these file by searching on the Web using the "DOS boot disk" key word.

Some of the USB flash storage manufacture provide utility to format and configure the USB flash to boot to DOS.

Due to the large pool of available USB flash storages; there is not a procedure or method to make the USB flash bootable to DOS that works for all USB flash in the market.

Some of the USB flash storage devices are built with file encryption, support multiple function and etc. Usually, these multi-function USB flash storage devices cannot be configured to boot to DOS. It's best to use a simple USB flash storage device that is 2GB or smaller.

In addition to formatting and configuring the USB flash storage to boot to DOS, you also need to copy the necessary DOS utilities to the USB flash storage and use them to format the SBC's IDE flash storage.

If the USB flash storage is made bootable to Windows Millennium or Windows 98, copy the following DOS utilities to the USB flash storage (make sure to copy the correct version of utility):

- Fdisk.exe
- Format.com
- Attrib.exe

If the USB flash storage is made bootable to DOS 6.22, in addition to the above utilities, the following are needed:

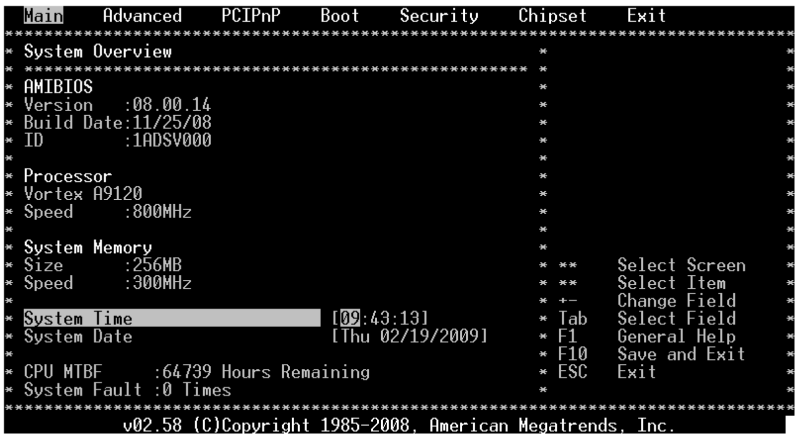
- Sys.com (Needed to transfer the DOS system file to SBC's IDE flash storage)
- Himem.sys (Extended memory driver needed for DOS 6.22 to support Loadcepc)

Configure the SBC to Boot from USB Flash Storage

Assuming you successfully prepared a bootable USB flash storage, insert the USB flash storage to the SBC and turn on the power. The USB flash storage must be inserted to the SBC's USB interface prior to power on, in order for the system BIOS to detect its present.

Improper BIOS settings affect system behavior and prevent the system from booting and function as expected.

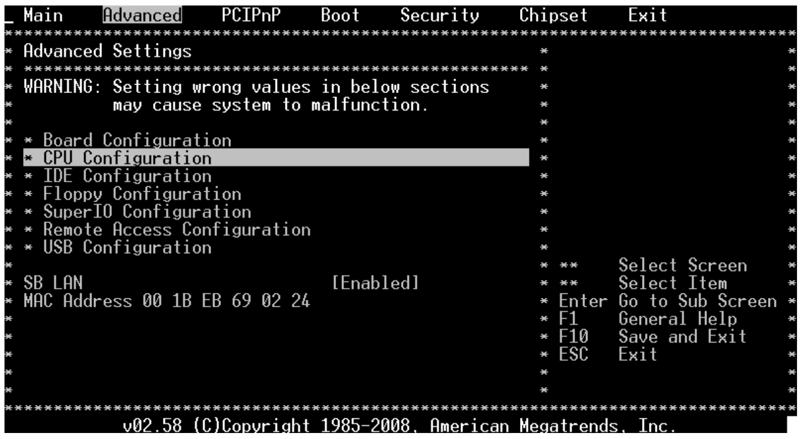
Immediately after power on, press the "Del" key repeatedly to bring up the BIOS configuration menu, similar to the screen in Figure K-1.



```
  Main    Advanced    PCIPnP    Boot    Security    Chipset    Exit
********************************************************************************
* System Overview                                 *                            *
* ************************************************ *                           *
* AMIBIOS                                         *                            *
* Version    :08.00.14                            *                            *
* Build Date:11/25/08                             *                            *
* ID         :1ADSV000                            *                            *
*                                                 *                            *
* Processor                                       *                            *
* Vortex A9120                                    *                            *
* Speed      :800MHz                              *                            *
*                                                 *                            *
* System Memory                                   *                            *
* Size       :256MB                               * **    Select Screen        *
* Speed      :300MHz                              * **    Select Item          *
*                                                 * +-    Change Field         *
* System Time                [09:43:13]           * Tab   Select Field         *
* System Date                [Thu 02/19/2009]     * F1    General Help         *
*                                                 * F10   Save and Exit        *
* CPU MTBF      :64739 Hours Remaining            * ESC   Exit                 *
* System Fault :0 Times                           *                            *
********************************************************************************
            v02.58 (C)Copyright 1985-2008, American Megatrends, Inc.
```

Figure K-1

Use the left/right arrow key to navigate to the Advanced settings menu, similar to Figure K-2.



```
  Main    Advanced    PCIPnP    Boot    Security    Chipset    Exit

 Advanced Settings

 WARNING: Setting wrong values in below sections
          may cause system to malfunction.

 * Board Configuration
 * CPU Configuration
 * IDE Configuration
 * Floppy Configuration
 * SuperIO Configuration
 * Remote Access Configuration
 * USB Configuration
                                                    **     Select Screen
 SB LAN                         [Enabled]           **     Select Item
 MAC Address 00 1B EB 69 02 24                      Enter  Go to Sub Screen
                                                    F1     General Help
                                                    F10    Save and Exit
                                                    ESC    Exit

           v02.58 (C)Copyright 1985-2008, American Megatrends, Inc.
```

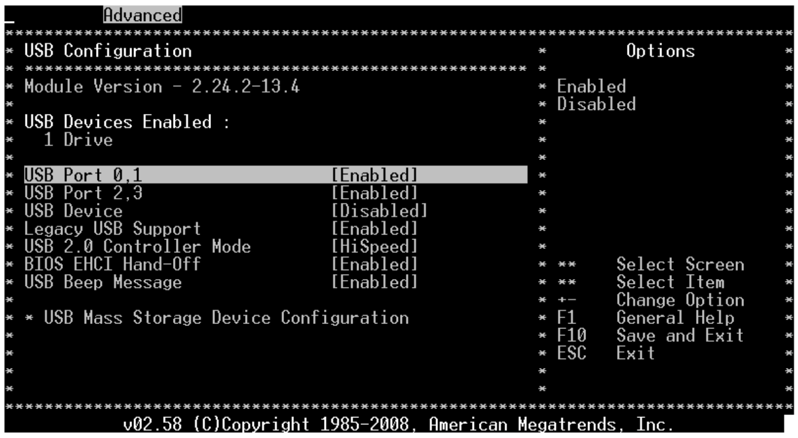
Figure K-2

The Advanced BIOS settings screen may not show the following options:

- Board Configuration
- Floppy Configuration
- SuperIO Configuration

These are hidden configuration.

Use the Up/Down arrow key to navigate to the “USB configuration” option and press Enter to bring up the USB configuration screen, as shown in Figure K-3.



```
          Advanced

 USB Configuration                                          Options

 Module Version - 2.24.2-13.4                       Enabled
                                                    Disabled
 USB Devices Enabled :
   1 Drive

 USB Port 0,1                   [Enabled]
 USB Port 2,3                   [Enabled]
 USB Device                     [Disabled]
 Legacy USB Support             [Enabled]
 USB 2.0 Controller Mode        [HiSpeed]
 BIOS EHCI Hand-Off             [Enabled]           **     Select Screen
 USB Beep Message               [Enabled]           **     Select Item
                                                    +-     Change Option
 * USB Mass Storage Device Configuration            F1     General Help
                                                    F10    Save and Exit
                                                    ESC    Exit

           v02.58 (C)Copyright 1985-2008, American Megatrends, Inc.
```

Figure K-3

Use the Up/Down arrow key to navigate to the “USB Mass Storage Device Configuration” and press enter to bring up the USB Mass Storage Device Configuration screen, as shown in Figure K-4.

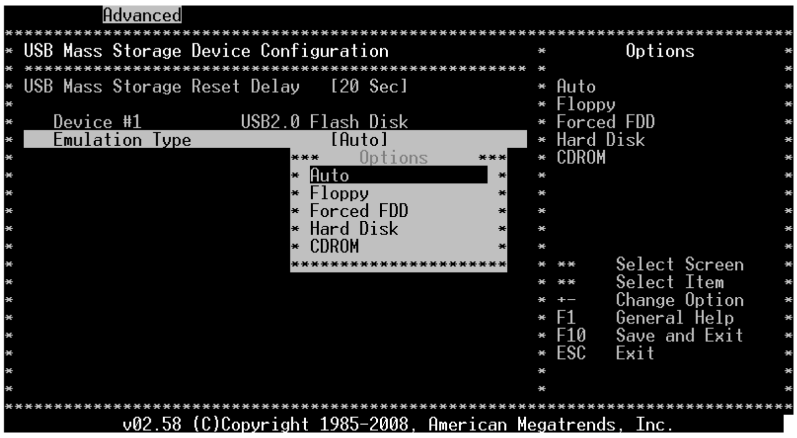

```
          Advanced
*************************************************************************
* USB Mass Storage Device Configuration         *        Options        *
* *********************************************  *                       *
* USB Mass Storage Reset Delay   [20 Sec]        * Auto                  *
*                                                * Floppy                *
*    Device #1          USB2.0 Flash Disk        * Forced FDD            *
*    Emulation Type              [Auto]          * Hard Disk             *
*                            ***   Options   *** * CDROM                 *
*                            * Auto            * *                       *
*                            * Floppy          * *                       *
*                            * Forced FDD      * *                       *
*                            * Hard Disk       * *                       *
*                            * CDROM           * *                       *
*                            ******************* * **    Select Screen   *
*                                                * **    Select Item     *
*                                                * +-    Change Option   *
*                                                * F1    General Help    *
*                                                * F10   Save and Exit   *
*                                                * ESC   Exit            *
*************************************************************************
            v02.58 (C)Copyright 1985-2008, American Megatrends, Inc.
```

Figure K-4

The “Device #1” setting list the detected USB flash storage device. The device name listed will be different for USB flash storage from different company.

The default Emulation Type is set to “Auto”, which works for some device and does not work for some device.

> You may have to go through multiple tries to find a setting that work.

For most USB flash storage device, setting the Emulation Type to “Hard Disk” yield the best result.

> With the USB flash storage emulation set to “Hard Disk”, the USB flash storage will boot up as a local hard drive.
>
> With the combination of the IDE flash storage (EmbedDisk) installed, the SBC will boot to DOS with two hard drives (C and D).
>
> You need to view the contents in both drive C and D to identify which drive letter has been assigned for the USB flash storage.

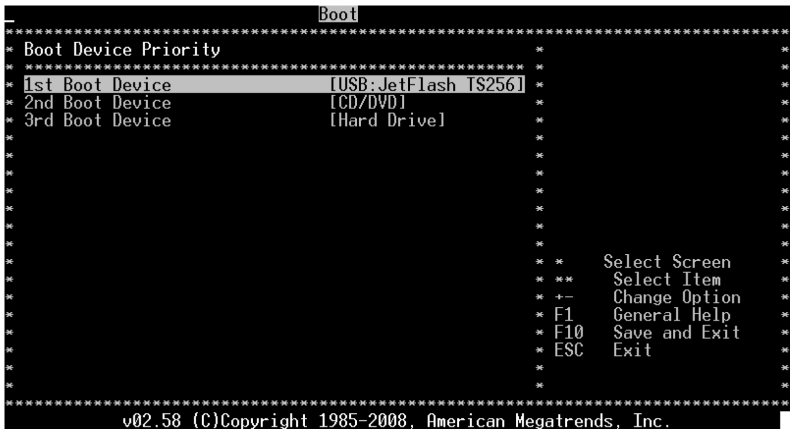
Boot Device Priority

In addition to configuring the USB flash storage device emulation type, the device boot sequence also affect the SBC’s boot process.

Press the “Esc” key to go back to the previous menu. You need to press the “Esc” a few times to go back to the main BIOS settings menu.

From the main BIOS setting menu, use the Left/Right arrow key to move to the Boot setting menu.

From the Boot Settings menu, use the Up/Down arrow key to navigate to the Boot Device Priority option and press enter to bring up the Boot Device Priority screen, as shown in Figure K-5.



```
                                   Boot
* Boot Device Priority                                      *
* ***********************************************************
* 1st Boot Device               [USB:JetFlash TS256]        *
* 2nd Boot Device               [CD/DVD]                    *
* 3rd Boot Device               [Hard Drive]                *

                                                            * *     Select Screen
                                                            * **     Select Item
                                                            * +-     Change Option
                                                            * F1     General Help
                                                            * F10    Save and Exit
                                                            * ESC    Exit

            v02.58 (C)Copyright 1985-2008, American Megatrends, Inc.
```

Figure K-5

There may only be two devices shown on the screen. Select the 1st Boot Device and press enter to bring up the Options screen, and select the USB flash storage device as the 1st Boot Device and press enter.

Press the “F10” key to save the settings and exit BIOS configuration.

The system will reset and boot from the USB flash storage.

Appendix L – *VDX-6326 SBC Jump Start CD-ROM*

The BSP, SDK and prebuilt OS runtime image for the jump start kit hardware, along with sample codes and utility are provided in the following directories on the CD-ROM:

Note: The files on the jump start CD-ROM are available for download from the following URL:
http://www.embeddedpc.net/Vortex86DXSPARK/

- \Eboot\
  The R6040 Ethernet boot loader for the VDX-6326 SBC is in this folder.
- \MSJK_Files\
  These are backup files from the root of the jump start kit hardware's flash storage. In the event the files on the SBC's flash storage are corrupted or deleted, recover the files from this folder.
- \SampleCodes\Application\
  The following sample Visual Studio projects are provided in this folder:
  - VB2005_SerialPortApplication.zip
    This is a simple serial port chat application written in Visual Basic 2005.
  - VS2005_HelloWorld.zip
    This is a Visual Studio 2005, C# hello world application.
  - VS2005_Win32.zip
    This is a Visual Studio 2005, Win32 hello world application.
  - VB2008_SerialPortApplication.zip
    This is a simple serial port chat application written in Visual Basic 2008.
  - VS2008_HelloWorld.zip
    This is a Visual Studio 2008, C# hello world application.
  - VS2008_Win32.zip
    This is a Visual Studio 2008, Win32 hello world application.
- \SampleCodes\GPIO\
  The following Vortex86DX GPIO sample projects are provided in the zip file:
  - GPIO - Visual Studio 2005 project for GPIO.dll, needed to access and use the VDX-6326 SBC's GPIO.
  - GPIO_CPP_Example - Visual Studio 2005 project showing how to access the GPIO.dll in native code.
  - GPIO_Net_Example - Visual Studio 2005 project showing how to access the GPIO.dll in managed code.
- \SampleCodes\OSDesign\
  Visual Studio 2005 project for MyWinCE OS design exercise.
- \SampleCodes\WatchdogTimer\
  The following Vortex86DX Watchdog Timers sample projects are provided in the zip file:
  - WDT - Visual Studio 2005 project for WDT.dll, needed to access and use the VDX-6326 SBC's Watchdog Timers.
  - WDT_CPP_Example - Visual Studio 2005 project showing how to access the WDT.dll in native code.
  - WDT_Net_Example - Visual Studio 2005 project showing how to access the WDT.dll in managed code.

- \Software\
  The following BSP, SDK and CE 6.0 components are provided in this folder:
    - AutoLaunch_v200_x86_WinCE600.msi
      When included in the CE 6.0 OS runtime, the AutoLaunch component can be configured to launch one or more application automatically when CE 6.0 OS starts.
    - CoreCon_v200_x86_WinCE600.msi
      The CoreCon component is needed to establish connectivity between the target device and development workstation's Visual Studio IDE to deploy application for testing and debug.
    - ICOP_VDX6326_60B_BSP.msi
      This is the BSP used for the exercise in this guide to develop the OS design.
    - RegFlushApp_v100_x86_WinCE600.msi
      This is a CE 6.0 utility used in the exercise in this guide to flush and save changed to the registry when Hive-based registry is implemented.
    - VDX6326_WINCE600_SDK.msi
      This SDK is generated from the MyWinCE OS design, to support application development for the VDX-6326 SBC using Visual Studio 2005 or 2008.

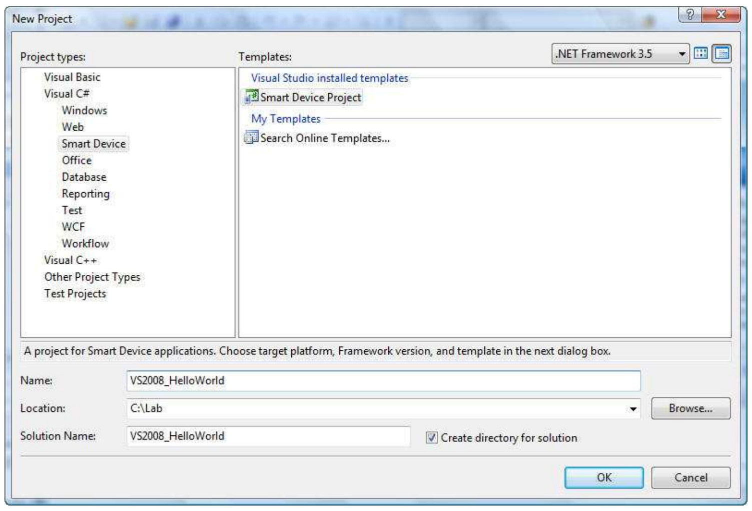
Appendix M – Visual Studio 2008 Managed Code Application

In this appendix, we will work through the steps to create a hello-world managed code application using VS2008. The following will be covered:

- Develop a C# managed code application using VS2008.
- Establish connection between the VS2008 development workstation and the target device using CoreCon.
- Download the application to the target device for testing and debug.

Step 1: Create a New Visual Studio 2008 C# Project

Launch VS2008, from VS2008 IDE, select **File | New | Project,** the following new project screen will appear, as shown in Figure M01.



New Project

Project types:
- Visual Basic
- Visual C#
  - Windows
  - Web
  - Smart Device
  - Office
  - Database
  - Reporting
  - Test
  - WCF
  - Workflow
- Visual C++
- Other Project Types
- Test Projects

Templates: .NET Framework 3.5

Visual Studio installed templates
- Smart Device Project

My Templates
- Search Online Templates...

A project for Smart Device applications. Choose target platform, Framework version, and template in the next dialog box.

Name: VS2008_HelloWorld

Location: C:\Lab

Solution Name: VS2008_HelloWorld

[✓] Create directory for solution

Fig. M01 - VS2008 IDE / New C# managed code Project

- On the Project type pane to the left, expand the Visual C# node and click on **Smart Device**.
- On the Templates pane to the right, select **Smart Device Project**.
- Enter **VS2008_HelloWorld** as the name for the project.
- Click on the **OK** button to bring up the “Add New Smart Device Project – VS2008_HelloWorld” screen, as shown in Figure M02.

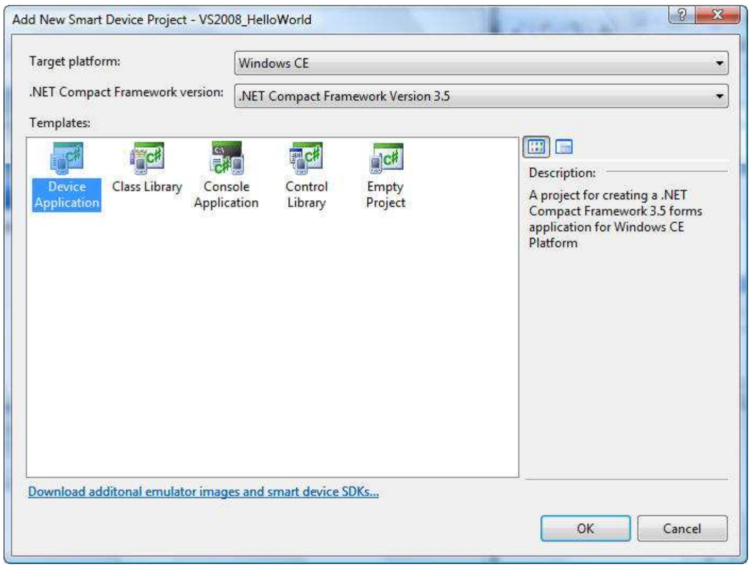

Add New Smart Device Project - VS2008_HelloWorld
Target platform:
Windows CE
.NET Compact Framework version:
.NET Compact Framework Version 3.5
Templates:
Device Application
Class Library
Console Application
Control Library
Empty Project
Description:
A project for creating a .NET Compact Framework 3.5 forms application for Windows CE Platform
Download additonal emulator images and smart device SDKs...
OK
Cancel

Fig. M02 - VS2008 IDE / New C# managed code Project

- Select **Windows CE** from the Target platform selection.
- Select **.NET Compact Framework Version 3.5** from the .NET Compact Framework version selection.
- Select **Device Application** from the Template selection.
- Click on the **OK** button to continue

The new project wizard will generate the initial project files, which include a blank form. Let's add some simple code to the application.

- Resize the Form1 to a smaller size (320x240) to make it easy to see the application when deploy on the CE 6.0 target device.
- Remove the mainMenu1 component placed onto Form1 by the wizard .
- Change the Form caption to "CE 6.0 Jump Start Kit – C# Example"
- Add a text-box to Form1 and change the name to textHelloWorld, clear the content in the text-box and place the text-box to the center of Form1.
- Add a button to Form1, change the name to buttonHelloWorld and change the text on the button's caption to "Hello World" and place the button to the center of Form1, just below the textHelloWorld text-box.

  Add the following code to the "buttonHelloWorld_Click" event.

```
textHelloWorld.Text = "Hello World!";
textHelloWorld.Text = "2nd Hello World!";
textHelloWorld.Text = "3rd Hello World!";
textHelloWorld.Text = "Last Hello World!";
```

From the VS2008 IDE, select **Build | Build Solution** to compile and generate executable binary for the project.

Step 2: Preparing Target Device to Connect to Visual Studio 2008

To perform this portion of the exercise, CE 6.0 image configured and built during the previous sections of this guide must be downloaded and running on the target device, and the VDX6326_WINCE600_SDK must be installed.

CoreCon is used to establish connectivity between the SBC and development workstation's VS2008 IDE.

To establish CoreCon connectivity, the following 5 CoreCon files need to be copied to the "\Windows" folder on the target device:

- Clientshutdown.exe
- ConmanClient2.exe
- CMaccept.exe
- eDbgTL.dll
- TcpConnectionA.dll

These files are installed to the following directory on the VS2008 development workstation, as part of the VS2008 installation:

```
\Program Files\Common Files\Microsoft Shared\CoreCon\1.0\Target\wce400\
```

CoreCon components supporting different type of processors are provided. There are multiple sub folders under the above directory, with name corresponding to supported processor. CoreCon files supporting the VDX-6326 SBC, built with x86 CPU, are in the \x86 sub folder.

> There are different versions of CoreCon files, installed to the development workstation as part of the Visual Studio 2005 and Visual Studio 2008 installation.
>
> CE 6.0 runtime image built with a different version of CoreCon files from the version on the development workstation will not be able to establish connectivity with the development workstation.

In the earlier section, during the customizing OS design steps, the CoreCon_v200_x86 component is added to the OS design. By adding this component, the same version of CoreCon files, installed to the development workstation as part of the Visual Studio installation, are included in the OS design and compiled as part of the CE 6.0 OS runtime image generated from the OS design.

To establish connectivity between the development workstation's VS2008 IDE and the SBC, we need to know the SBC's IP address. Work through the following steps to find the IP address for the SBC:

- With CE 6.0 running on the target device, click on **Start | Run** and launch the **CMD** command to open a console command window, as shown in Figure M03.

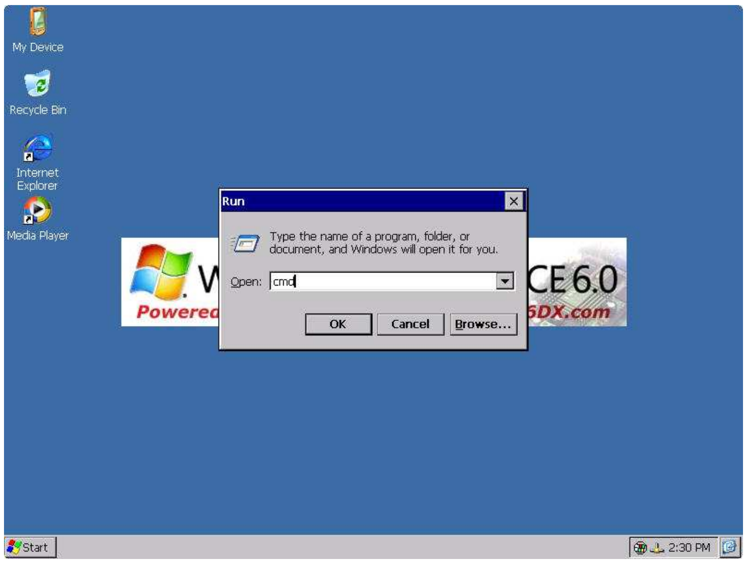
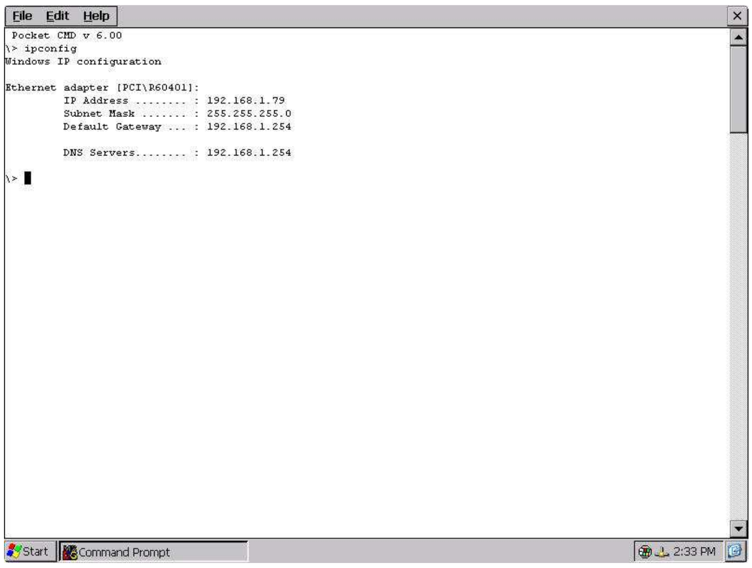


Fig. M03 - CE 6.0 desktop – Launching the CMD command

- From within the console command window, type **IpConfig** to show the IP address for the target device, as shown in Figure M04.

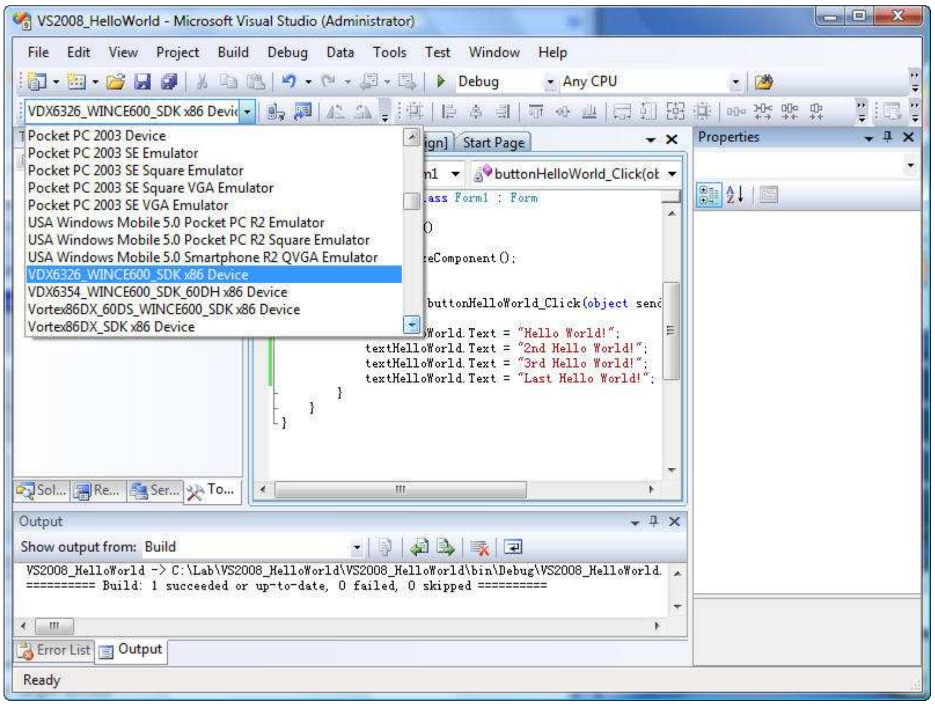


```
 Pocket CMD v 6.00
\> ipconfig
Windows IP configuration

Ethernet adapter [PCI\R60401]:
        IP Address ........ : 192.168.1.79
        Subnet Mask ....... : 255.255.255.0
        Default Gateway ... : 192.168.1.254

        DNS Servers........ : 192.168.1.254

\> 
```

Fig. M04 - CE 6.0 console command window showing the IP address information

Now that we have the IP address for the target device, let’s move to VS2008 IDE to configure device settings. In order for this to work, both the VS2008 development workstation and target device must be connected to the same LAN segment and acquire their IP address from the same DHCP server.

From the VS2008 IDE, set the target device to “**VDX6326_WINCE600_SDK x86 Device**”, as shown in Figure M05.



VS2008_HelloWorld - Microsoft Visual Studio (Administrator)

VDX6326_WINCE600_SDK x86 Device

- Pocket PC 2003 Device
- Pocket PC 2003 SE Emulator
- Pocket PC 2003 SE Square Emulator
- Pocket PC 2003 SE Square VGA Emulator
- Pocket PC 2003 SE VGA Emulator
- USA Windows Mobile 5.0 Pocket PC R2 Emulator
- USA Windows Mobile 5.0 Pocket PC R2 Square Emulator
- USA Windows Mobile 5.0 Smartphone R2 QVGA Emulator
- VDX6326_WINCE600_SDK x86 Device
- VDX6354_WINCE600_SDK_60DH x86 Device
- Vortex86DX_60DS_WINCE600_SDK x86 Device
- Vortex86DX_SDK x86 Device

```
                textHelloWorld.Text = "2nd Hello World!";
                textHelloWorld.Text = "3rd Hello World!";
                textHelloWorld.Text = "Last Hello World!";
            }
        }
}
```

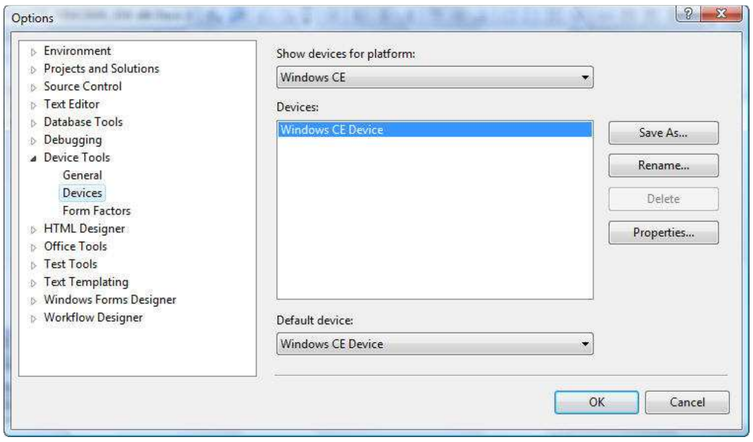
Output

Show output from: Build

```
VS2008_HelloWorld -> C:\Lab\VS2008_HelloWorld\VS2008_HelloWorld\bin\Debug\VS2008_HelloWorld.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Fig. M05 - VS2008 IDE / select target device

From VS2008 IDE, select **Tools | Options…** to bring up the following configuration screen, as shown in Figure M06.



Options

- Environment
- Projects and Solutions
- Source Control
- Text Editor
- Database Tools
- Debugging
- Device Tools
  - General
  - Devices
  - Form Factors
- HTML Designer
- Office Tools
- Test Tools
- Text Templating
- Windows Forms Designer
- Workflow Designer

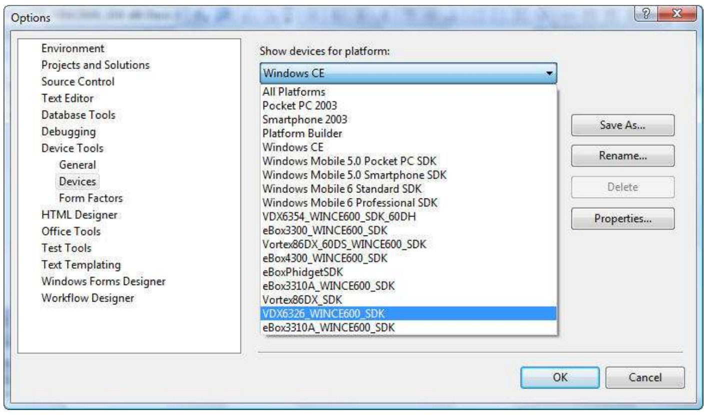
Show devices for platform: Windows CE

Devices: Windows CE Device

Default device: Windows CE Device

Fig. M06 - VS2008 Tools Options

- On the left, click to expand the **Device Tools** node and click on the **Devices** node.
- On the right, select **VDX6326_WINCE600_SDK** from the list of available platform in the **Show devices for platform** combo text box, and select **VDX6326_WINCE600_SDK x86 Device** as the default device, as shown in Figure M07.


Options

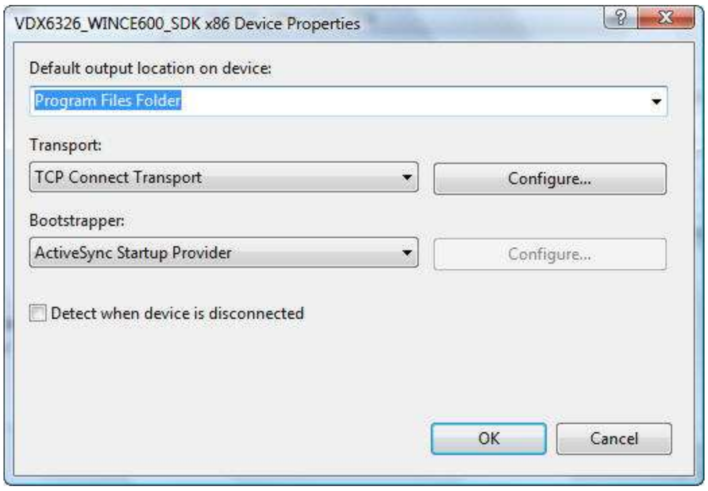
Environment
Projects and Solutions
Source Control
Text Editor
Database Tools
Debugging
Device Tools
General
Devices
Form Factors
HTML Designer
Office Tools
Test Tools
Text Templating
Windows Forms Designer
Workflow Designer

Show devices for platform:
Windows CE

All Platforms
Pocket PC 2003
Smartphone 2003
Platform Builder
Windows CE
Windows Mobile 5.0 Pocket PC SDK
Windows Mobile 5.0 Smartphone SDK
Windows Mobile 6 Standard SDK
Windows Mobile 6 Professional SDK
VDX6354_WINCE600_SDK_60DH
eBox3300_WINCE600_SDK
Vortex86DX_60DS_WINCE600_SDK
eBox4300_WINCE600_SDK
eBoxPhidgetSDK
eBox3310A_WINCE600_SDK
Vortex86DX_SDK
VDX6326_WINCE600_SDK
eBox3310A_WINCE600_SDK

Fig. M07 - VS2008 Tools Options

- Click on the **Properties** button to bring up **VDX6326_WINCE600_SDK x86 Device Properties** setting screen, as shown in Figure M08.


VDX6326_WINCE600_SDK x86 Device Properties

Default output location on device:
Program Files Folder

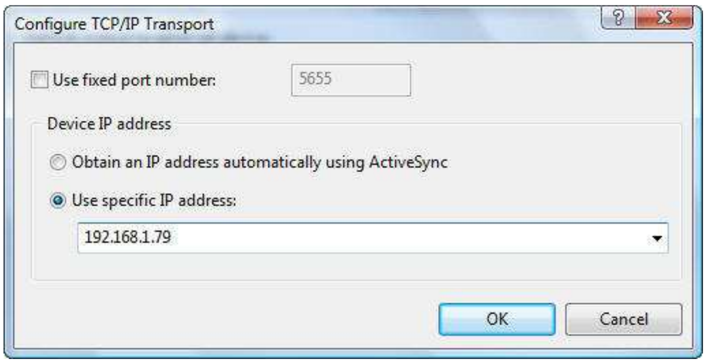
Transport:
TCP Connect Transport

Bootstrapper:
ActiveSync Startup Provider

Detect when device is disconnected

Fig. M08 - VDX6326_WINCE600 x86 Device properties

- Click on the **Configure** button to bring up **Configure TCP/IP Transport** screen.
- Select **Use specific IP address** and enter IP address for the target device, as shown in Figure M09.


Configure TCP/IP Transport

Use fixed port number: 5655

Device IP address

Obtain an IP address automatically using ActiveSync

Use specific IP address:
192.168.1.79

Fig. M09 - Configure TCP/IP Transport / Set device IP Address

- Click the **OK** button to save device IP address setting.
- Click the **OK** button on the **VDX6326_WINCE600_SDK x86 Device Properties** screen to close the screen.
- Click the **OK** button on the **Options** screen to close the screen.

Step 3: Connecting Target Device to Visual Studio 2008 IDE using CoreCon

To initiate connectivity between development workstation's VS2008 IDE and target device using CoreCon, we need to launch the **ConmanClient2.exe** and **cMaccept.exe** CoreCon components from the target device, with CE 6.0 running. This is a cumbersome process and has to perform at least once for each time the target device reset power.

To make it easier to establish CoreCon connectivity, CoreCon files were added to the OS design and included to the OS runtime image. During customizing the OS design step, we added the CoreCon_v200_x86 component, AutoLaunch_v200_x86 component and entered registry entries to launch the CoreCon component automatically when the CE 6.0 OS starts.

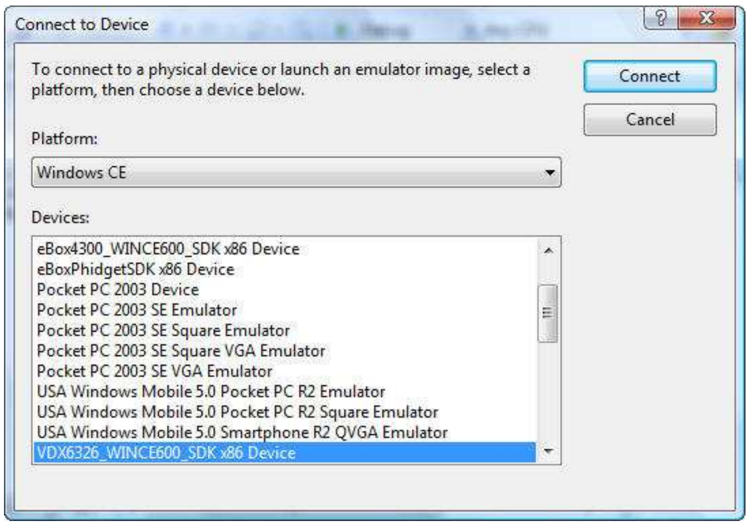
> The CoreCon_v200_x86 component is used for development purpose and should not be included in the release image, and should be removed from the image intended for distribution.

With the CoreCon_v200_x86 and AutoLaunch_v200_x86 components already included in the OS runtime image, along with the registry configured to launch CoreCon automatically when the CE 6.0 OS starts, the **Conmanclient2.exe** executable is launched automatically each time the OS runtime image starts.

> The following registry entries were added to the OS design by the CoreCon_v200_x86 component, to override system security and enable CoreCon connectivity to be established without the need to launch the **cMaccept.exe** executable:
>
> ```
> [HKEY_LOCAL_MACHINE\System]
>       "CoreConOverrideSecurity"=dword:1
> ```
>
> Without the above registry, in addition to ConmanClient2.exe, you need to launch cMaccept.exe executable to establish CoreCon connectivity.
>
> The cMaccept.exe executable needs to launch after ConmanClient2.exe. When executed, the cMaccept.exe disables the system security temporary to establish CoreCon connectivity, and will time-out in 3 minutes. If CoreCon connectivity is not established within 3 minutes, you need to launch cMaccept.exe again to establish CoreCon connectivity.

With ConmanClient2.exe automatically launched when the CE 6.0 OS runtime starts, work through the following steps to establish CoreCon connectivity between the development workstation's VS2008 IDE and the target device:

- From VS2008 IDE, select **Tools | Connect to device…** and select **VDX6326_WINCE600_SDK** from the list of available devices, and click on the **Connect** button, as shown in Figure M10.


Connect to Device

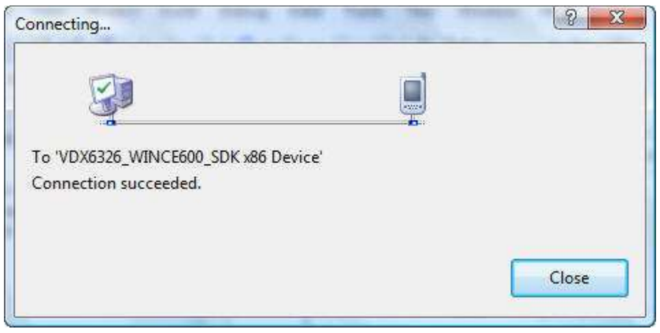
To connect to a physical device or launch an emulator image, select a platform, then choose a device below.

Platform: Windows CE

Devices:
- eBox4300_WINCE600_SDK x86 Device
- eBoxPhidgetSDK x86 Device
- Pocket PC 2003 Device
- Pocket PC 2003 SE Emulator
- Pocket PC 2003 SE Square Emulator
- Pocket PC 2003 SE Square VGA Emulator
- Pocket PC 2003 SE VGA Emulator
- USA Windows Mobile 5.0 Pocket PC R2 Emulator
- USA Windows Mobile 5.0 Pocket PC R2 Square Emulator
- USA Windows Mobile 5.0 Smartphone R2 QVGA Emulator
- VDX6326_WINCE600_SDK x86 Device

Fig. M10 - Connect to device

When connection is successful, the Connecting dialog box will display Connection succeeded to indicate a successful connection, as shown in Figure M11.


Connecting...

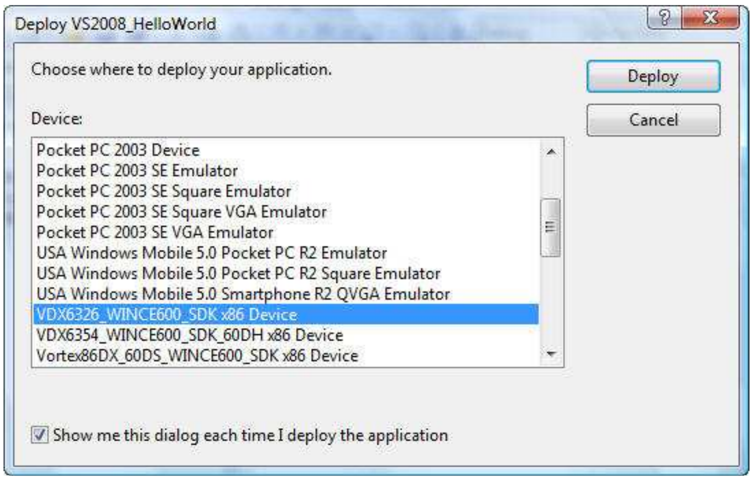
To 'VDX6326_WINCE600_SDK x86 Device'
Connection succeeded.

Fig. M11 - Target device connected

Step 4: Deploy C# Application to Target Device

With the CoreCon Connectivity established, we are ready to deploy the C# application to the target device.

From VS2008 IDE, select **Debug | Start Debugging** to bring up the **Deploy VS2008_HelloWorld** screen, as shown in Figure M12.


Deploy VS2008_HelloWorld

Choose where to deploy your application.

Device:

Pocket PC 2003 Device
Pocket PC 2003 SE Emulator
Pocket PC 2003 SE Square Emulator
Pocket PC 2003 SE Square VGA Emulator
Pocket PC 2003 SE VGA Emulator
USA Windows Mobile 5.0 Pocket PC R2 Emulator
USA Windows Mobile 5.0 Pocket PC R2 Square Emulator
USA Windows Mobile 5.0 Smartphone R2 QVGA Emulator
VDX6326_WINCE600_SDK x86 Device
VDX6354_WINCE600_SDK_60DH x86 Device
Vortex86DX_60DS_WINCE600_SDK x86 Device

☑ Show me this dialog each time I deploy the application

Fig. M12 - Deploy VS2008_HelloWorld C# application

- Select **VDX6326_WINCE600_SDK** and click **Deploy**
- The C# managed code application will download and launch on target device as shown in the following screen shot, Figure M13.

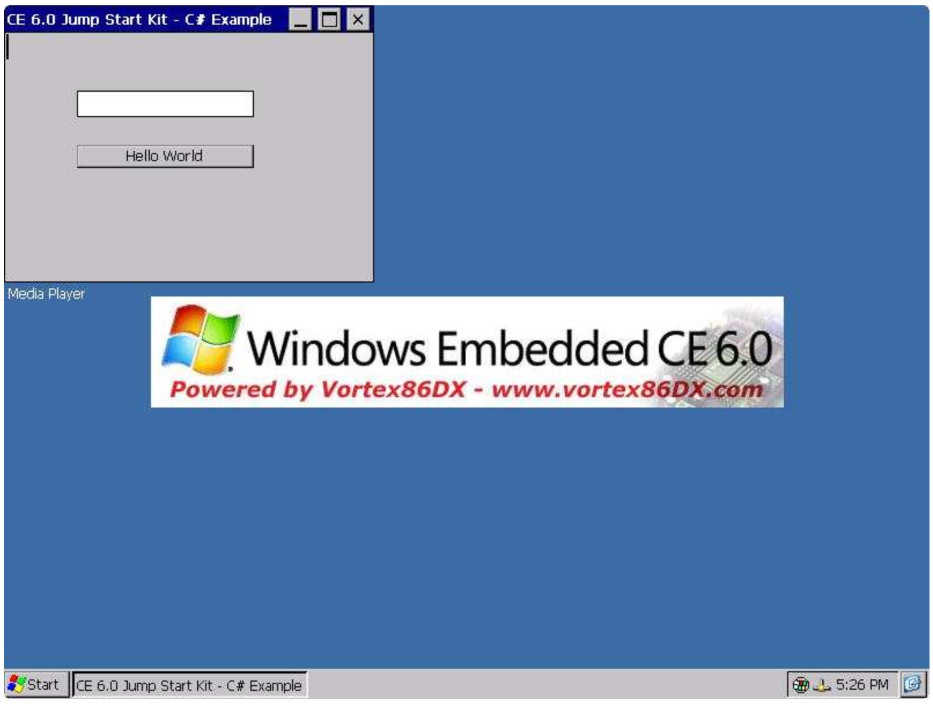

CE 6.0 Jump Start Kit - C# Example
Hello World
Media Player
Windows Embedded CE 6.0
Powered by Vortex86DX - www.vortex86DX.com
Start
CE 6.0 Jump Start Kit - C# Example
5:26 PM

Fig. M13 - CE 6.0 desktop with C# managed code application running

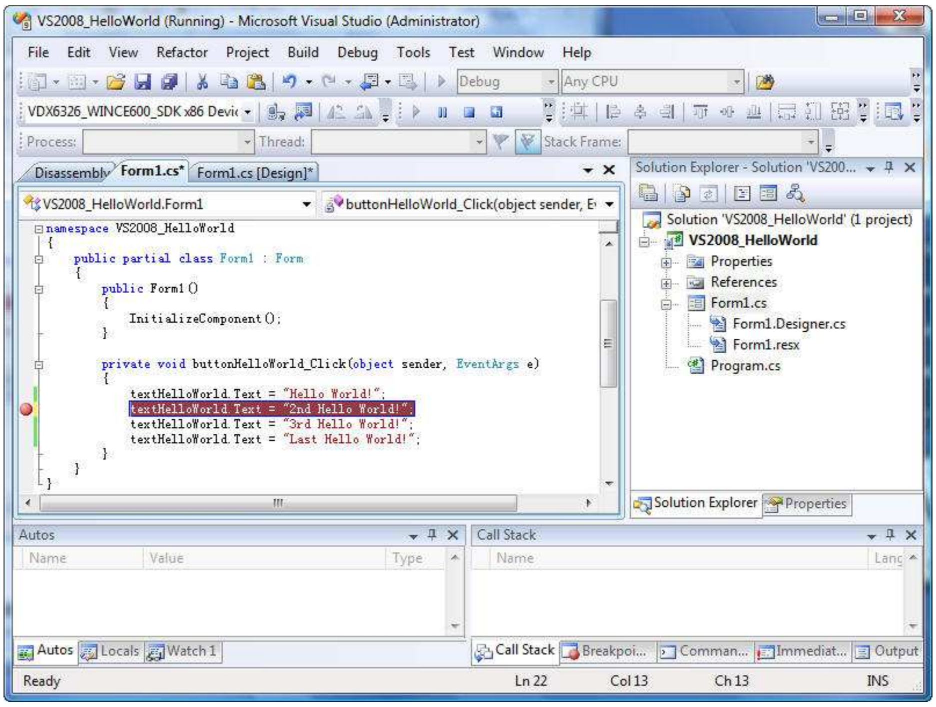
Step 5: Debug the C# Application running on Target Device

While the VS2008_HelloWorld C# application is running on the target device, we can set breakpoint to the source codes to halt the application when the application execution reaches the breakpoint.

Work through the following steps to set a breakpoint in the `buttonHelloWorld_Click` event:

- From the VS2008 IDE, navigate to the buttonHelloWorld_Click event code segment, click on the following line of code and press the F9 key to set a breakpoint, as shown in Figure M14.

```
textHelloWorld.Text = "2nd Hello World!";
```



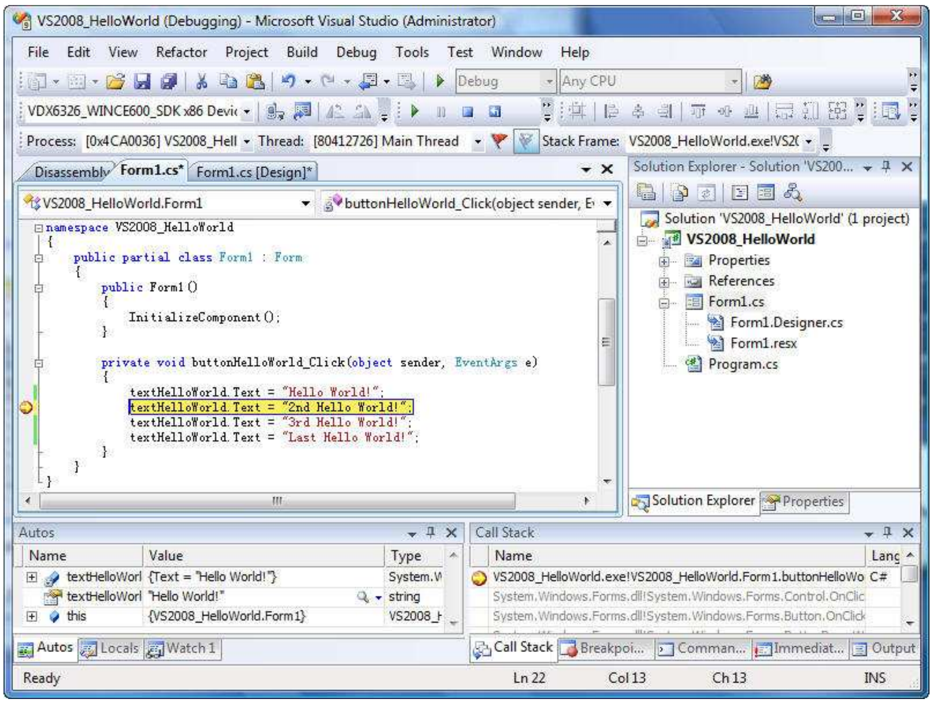
```
namespace VS2008_HelloWorld
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void buttonHelloWorld_Click(object sender, EventArgs e)
        {
            textHelloWorld.Text = "Hello World!";
            textHelloWorld.Text = "2nd Hello World!";
            textHelloWorld.Text = "3rd Hello World!";
            textHelloWorld.Text = "Last Hello World!";
        }
    }
}
```

Fig. M14 - Setting breakpoint

The above breakpoint is set while the VS2008_HelloWorld application is running on the target device.

With the breakpoint in place, work through the following steps to execute the VS2008_HelloWorld application, reaching the breakpoint and halt.

- From the target device's CE 6.0 desktop, with the VS2008_HelloWorld application running, click on the **Hello World** button.
- From the VS2008 IDE on the development workstation, the breakpoint highlight color changed to **yellow**, an indication the program is halt at this line of code, as shown in Figure M15.


VS2008_HelloWorld (Debugging) - Microsoft Visual Studio (Administrator)
File Edit View Refactor Project Build Debug Tools Test Window Help
Debug
Any CPU
VDX6326_WINCE600_SDK x86 Devic
Process: [0x4CA0036] VS2008_Hell
Thread: [80412726] Main Thread
Stack Frame: VS2008_HelloWorld.exe!VS2(
Disassembly
Form1.cs*
Form1.cs [Design]*
VS2008_HelloWorld.Form1
buttonHelloWorld_Click(object sender, E
Solution Explorer - Solution 'VS200...
Solution 'VS2008_HelloWorld' (1 project)
VS2008_HelloWorld
Properties
References
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
Solution Explorer
Properties
Autos
Locals
Watch 1
Call Stack
Breakpoi...
Comman...
Immediat...
Output
Ready
Ln 22
Col 13
Ch 13
INS

```
namespace VS2008_HelloWorld
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void buttonHelloWorld_Click(object sender, EventArgs e)
        {
            textHelloWorld.Text = "Hello World!";
            textHelloWorld.Text = "2nd Hello World!";
            textHelloWorld.Text = "3rd Hello World!";
            textHelloWorld.Text = "Last Hello World!";
        }
    }
}
```

Autos

Name	Value	Type
textHelloWorl	{Text = "Hello World!"}	System.W
textHelloWorl	"Hello World!"	string
this	{VS2008_HelloWorld.Form1}	VS2008_H

Call Stack

Name	Lang
VS2008_HelloWorld.exe!VS2008_HelloWorld.Form1.buttonHelloWo	C#
System.Windows.Forms.dll!System.Windows.Forms.Control.OnClic	
System.Windows.Forms.dll!System.Windows.Forms.Button.OnClick	

Fig. M15 - Execution halt at the breakpoint

- The application is running on the target device and is halt on the following line of code.

```
textHelloWorld.Text = "2nd Hello World!";
```

  As the code execution is halt on the above line of code, the code has not been executed. The textbox on the VS2008_HelloWorld application screen is showing the “Hello World!” message, indicating the line of code just before the breakpoint has been executed.

- From the VS2008 IDE, press the F11 key to step through one line of code.

As we press the F11 key, we can see the next line of code becomes highlighted with yellow. The textbox on the VS2008_HelloWorld application running on the target device is changed to “2nd Hello World!”.

- The F11 key is used to step through the code one line at a time. To continue the code execution, press the F5 key from the VS2008 IDE.

From the VS2008_HelloWorld application screen, running on the target device, you can click on the **Hello World** button again to execute the code and reach the breakpoint to halt the execution.

As you can see from the simple exercise in this section, CE 6.0 and the Visual Studio development environment provide an effective, efficient and easy to use development environment.

Appendix N – Visual Studio 2008 Native Code Application

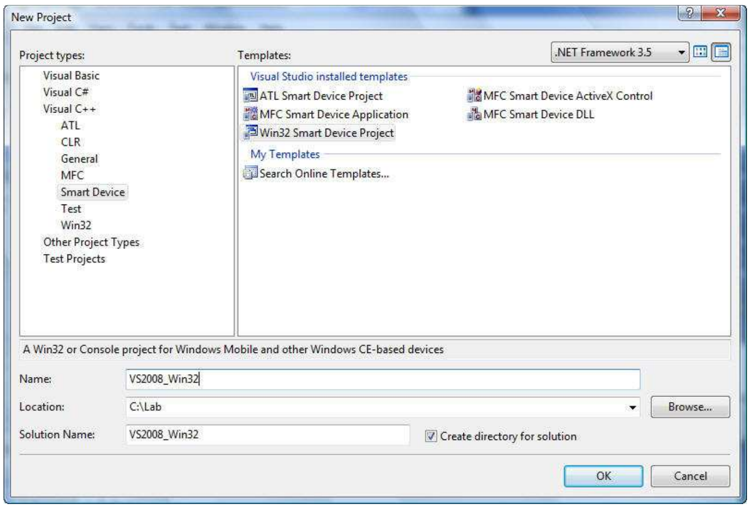
In this section, we will cover the following:

- Develop a simple Win32 native code application using VS2008.
- Establish connection between the development workstation and target device using CoreCon.
- Download the application to the target device with the CE 6.0 OS runtime image built in the earlier section.

To work through the exercise in this section, you need to install the SDK generated from the OS design in the earlier section, VDX6326_WINCE600_SDK.msi.

Step 1: Create a New Visual Studio 2008 C++ Project

From VS2008 IDE, Select **File | New | Project,** the following new project screen will appear, as shown in Figure N01.



A Win32 or Console project for Windows Mobile and other Windows CE-based devices

Name: VS2008_Win32

Location: C:\Lab

Solution Name: VS2008_Win32

☑ Create directory for solution

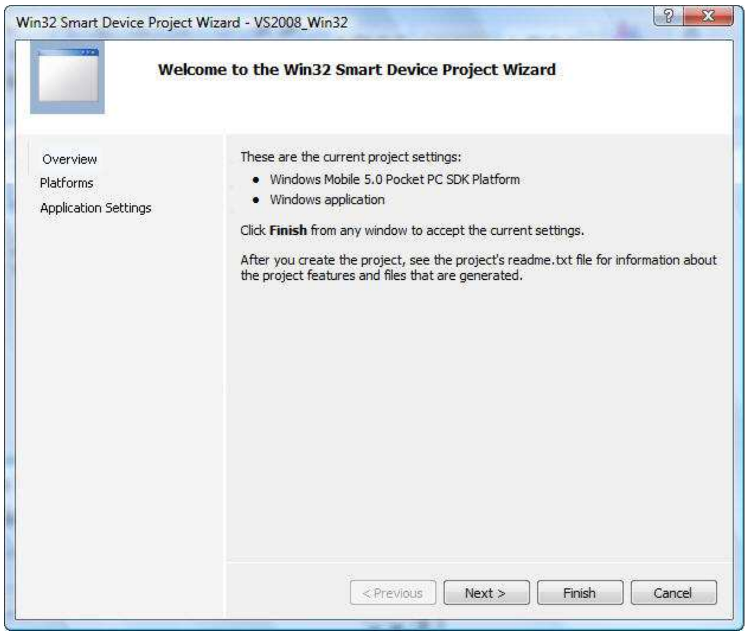
Fig. N01- VS2008 IDE / New Win32 native code Project

- On the new project wizard screen's left pane, expand **Visual C++** node and select **Smart Device** as the project type. On the right pane, select **Win32 Smart Device Project**.
- Enter **VS2008_Win32** as the name of the project.
- Enter **C:\Lab** as the location of the project.

You can choose a different folder to place the VS2008_Win32 project.

- Click on the **OK** button to bring up the Win32 Smart Device Project wizard, as shown in Figure N02.

The new project wizard will create the VS2008_Win32 project folder under the C:\Lab directory.



Win32 Smart Device Project Wizard - VS2008_Win32

Welcome to the Win32 Smart Device Project Wizard

These are the current project settings:

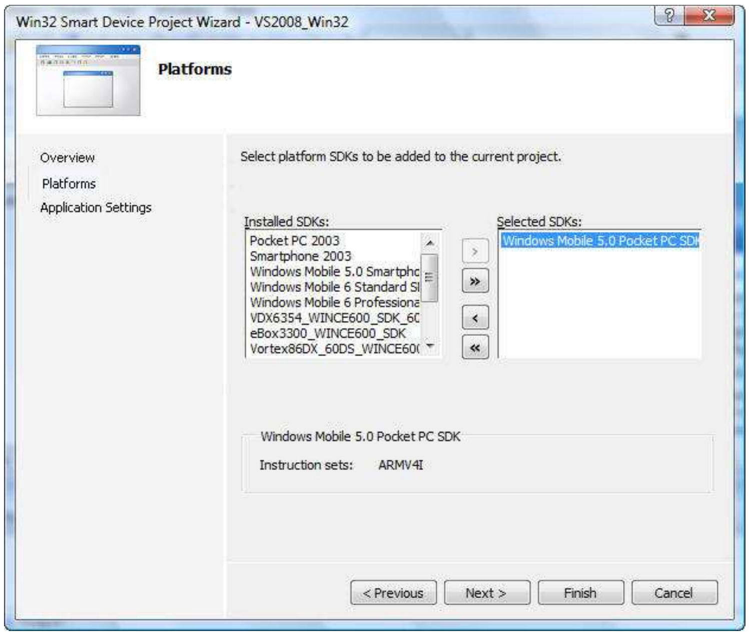
- Windows Mobile 5.0 Pocket PC SDK Platform
- Windows application

Click **Finish** from any window to accept the current settings.

After you create the project, see the project's readme.txt file for information about the project features and files that are generated.

Fig. N02- Win32 Smart Device Application wizard

- Click on the Next button to bring up the Platforms selection step, as shown in Figure N03.



Win32 Smart Device Project Wizard - VS2008_Win32

Platforms

Select platform SDKs to be added to the current project.

Installed SDKs:
Pocket PC 2003
Smartphone 2003
Windows Mobile 5.0 Smartphc
Windows Mobile 6 Standard SI
Windows Mobile 6 Professiona
VDX6354_WINCE600_SDK_6C
eBox3300_WINCE600_SDK
Vortex86DX_60DS_WINCE60(

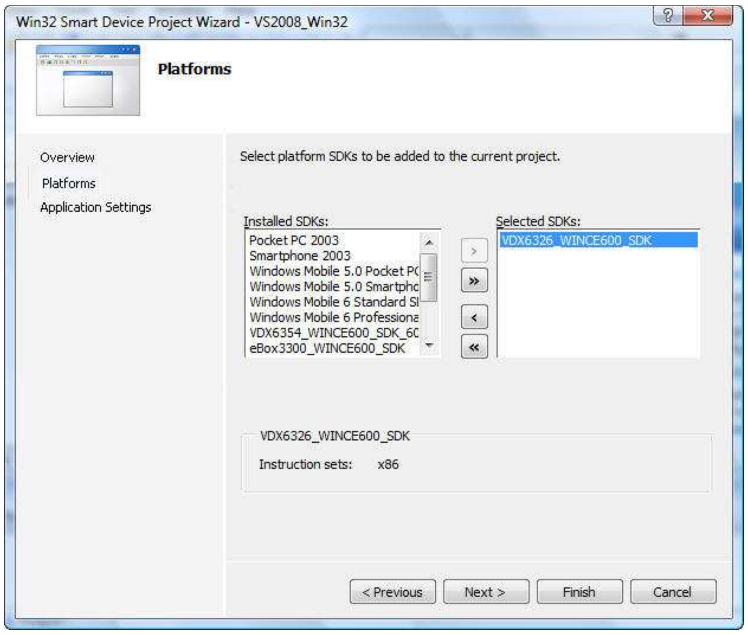
Selected SDKs:
Windows Mobile 5.0 Pocket PC SDK

Windows Mobile 5.0 Pocket PC SDK
Instruction sets: ARMV4I

Fig. N03- Platform and SDK selection

- From the Selected SDKs pane on the right, click to highlight the default entry, **Windows Mobile 5.0 Pocket PC SDK**.
- Click on the single left pointing arrow, located to the left of the Selected SDKs pane to remove the **Windows Mobile 5.0 Pocket PC SDK** entry from the selected SDKs pane.

- From the Installed SDKs pane on the left, click to highlight the **VDX6326_WINCE600_SDK** entry.
- Click on the right pointing arrow, located to the right of the Installed SDKs pane, to add the **VDX6326_WINCE600_SDK** to the selected SDKs pane, as shown in Figure N04



Select platform SDKs to be added to the current project.

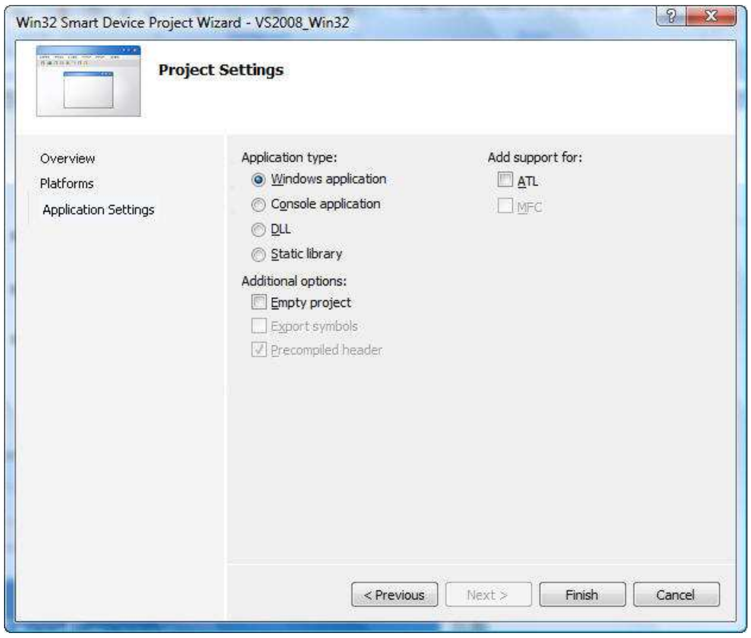
Installed SDKs:
Pocket PC 2003
Smartphone 2003
Windows Mobile 5.0 Pocket PC
Windows Mobile 5.0 Smartphc
Windows Mobile 6 Standard SI
Windows Mobile 6 Professiona
VDX6354_WINCE600_SDK_6C
eBox3300_WINCE600_SDK

Selected SDKs:
VDX6326_WINCE600_SDK

VDX6326_WINCE600_SDK
Instruction sets: x86

Fig. N04 - Platform and SDK selection

- Click on the **Next** button to bring up the Project Settings step, as shown in Figure N05.



Application type:
(●) Windows application
( ) Console application
( ) DLL
( ) Static library

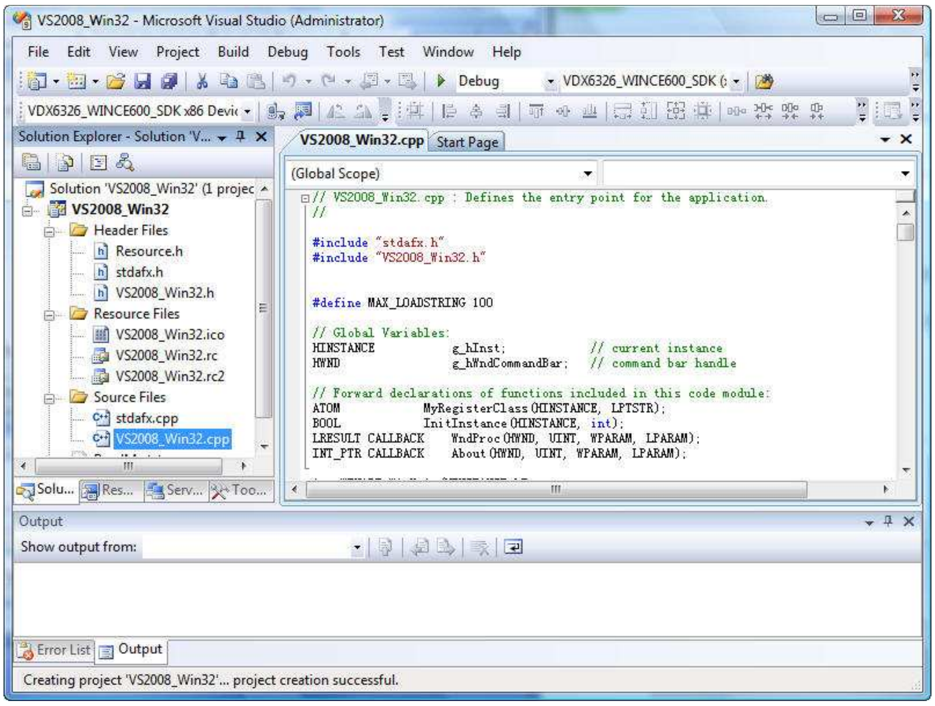
Additional options:
[ ] Empty project
[ ] Export symbols
[✓] Precompiled header

Add support for:
[ ] ATL
[ ] MFC

Fig. N05 - Project settings

- Keep the default selection to create a **Windows application**.
- Click on the **Finish** button to complete the wizard and generate the initial project files.

After the project wizard step is completed, the VS2008_Win32 project files are created in the C:\Lab\ VS2008_Win32 folder. With the VS2008_Win32 project active, the VS2008 IDE should look similar to IDE as shown in Figure N06.



```
// VS2008_Win32.cpp : Defines the entry point for the application.
//

#include "stdafx.h"
#include "VS2008_Win32.h"


#define MAX_LOADSTRING 100

// Global Variables:
HINSTANCE           g_hInst;            // current instance
HWND                g_hWndCommandBar;   // command bar handle

// Forward declarations of functions included in this code module:
ATOM            MyRegisterClass(HINSTANCE, LPTSTR);
BOOL            InitInstance(HINSTANCE, int);
LRESULT CALLBACK    WndProc(HWND, UINT, WPARAM, LPARAM);
INT_PTR CALLBACK    About(HWND, UINT, WPARAM, LPARAM);
```

Creating project 'VS2008_Win32'... project creation successful.

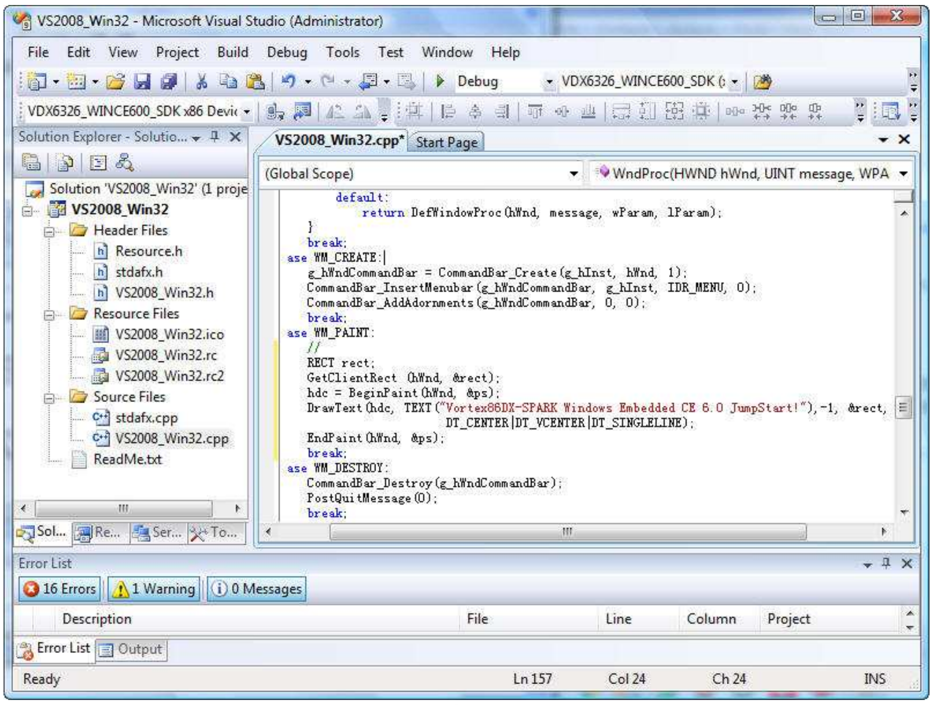
Fig. N06 - VS2008 IDE with VS2008_Win32 project active

Work through the following steps to add some code to the VS2008_Win32 project.

- From the VS2008 IDE's Solution Explorer tab, double click on the **VS2008_Win32.cpp** source file, in the `\Source` Files folder to view and edit the codes in the center code editor window.

- Replace the codes in the "**case WM_PAINT:**" statement with the following codes, as shown in Figure N07.

```
Case WM_PAINT:
    //
    RECT rect;
    GetClientRect (hWnd, &rect);
    hdc = BeginPaint(hWnd, &ps);
    DrawText(hdc, TEXT("Vortex86DX-SPARK Windows Embedded CE 6.0 JumpStart!"), -1, &rect,
                        DT_CENTER|DT_VCENTER|DT_SINGLELINE);
    EndPaint(hWnd. &ps);
    break;
```



```
            default:
                return DefWindowProc(hWnd, message, wParam, lParam);
        }
        break;
ase WM_CREATE:
        g_hWndCommandBar = CommandBar_Create(g_hInst, hWnd, 1);
        CommandBar_InsertMenubar(g_hWndCommandBar, g_hInst, IDR_MENU, 0);
        CommandBar_AddAdornments(g_hWndCommandBar, 0, 0);
        break;
ase WM_PAINT:
        //
        RECT rect;
        GetClientRect (hWnd, &rect);
        hdc = BeginPaint(hWnd, &ps);
        DrawText(hdc, TEXT("Vortex86DX-SPARK Windows Embedded CE 6.0 JumpStart!"),-1, &rect,
                              DT_CENTER|DT_VCENTER|DT_SINGLELINE);
        EndPaint(hWnd, &ps);
        break;
ase WM_DESTROY:
        CommandBar_Destroy(g_hWndCommandBar);
        PostQuitMessage(0);
        break;
```

Fig. N07 - VS2008 IDE with VS2008_Win32 project active.

- From the VS2008 IDE, select **Build | Build Solution** to build the VS2008_Win32 project.

Step 2: Preparing Target Device to Connect to Visual Studio 2008 IDE

To perform this portion of the exercise, CE 6.0 OS runtime image configured and built during the previous sections of this guide must be downloaded and running on the target device.

We will use CoreCon to establish the connection between the VS2008 IDE and the target device.

The following 5 CoreCon files need to be copied to `\Windows` folder on the target device.

- Clientshutdown.exe
- ConmanClient2.exe
- CMaccept.exe
- eDbgTL.dll
- TcpConnectionA.dll

These files are installed to the following directory on the VS2008 development workstation, as part of the VS2008 installation.

```
\Program Files\Common Files\Microsoft Shared\CoreCon\1.0\Target\wce400\
```

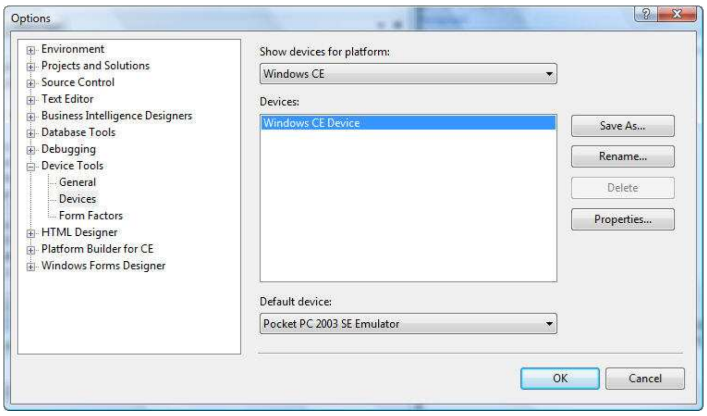
CoreCon components supporting different CPU architectures are provided. There are multiple sub folders under the above directory, with names corresponding to the each CPU family, containing the CoreCon files for each CPU family. CoreCon files supporting the target device, built with x86 CPU, are in the \x86 sub folder.

In the earlier section, during the customizing OS design steps, the CoreCon_v200_x86 component is added to the OS design. By adding this component, the CoreCon files are included in the OS design and compiled as part of the CE 6.0 OS runtime image generated from the OS design.

To establish connection between the VS2008 development workstation and the target device, we need to know the target device's IP address. For this exercise, the target device is configured with a static IP address, 192.168.2.232 with 255.255.255.0 as the subnet. The development workstation is configured with a static IP address in the same subnet, 192.168.2.132.

A cross-over RJ45 Ethernet cable is used to connect the target device directly to the development workstation. Work through the following steps to establish connectivity between the target device and VS2008 IDE:

- From VS2008 IDE, select **Tools | Options…** to bring up the following configuration screen, as shown in Figure N08.



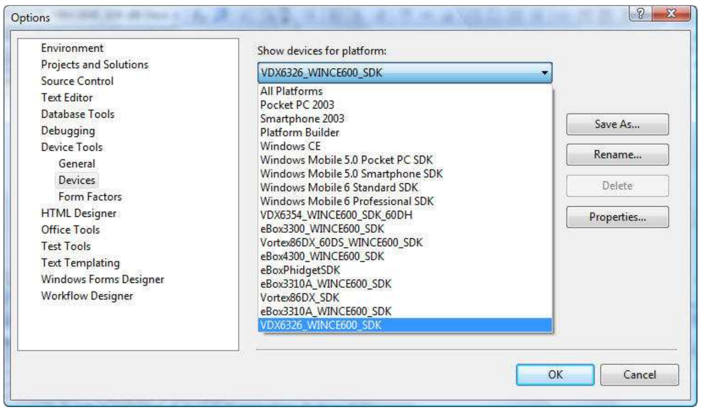
Show devices for platform: Windows CE

Devices: Windows CE Device

Default device: Pocket PC 2003 SE Emulator

Fig. N08 - VS2008 Tools Options

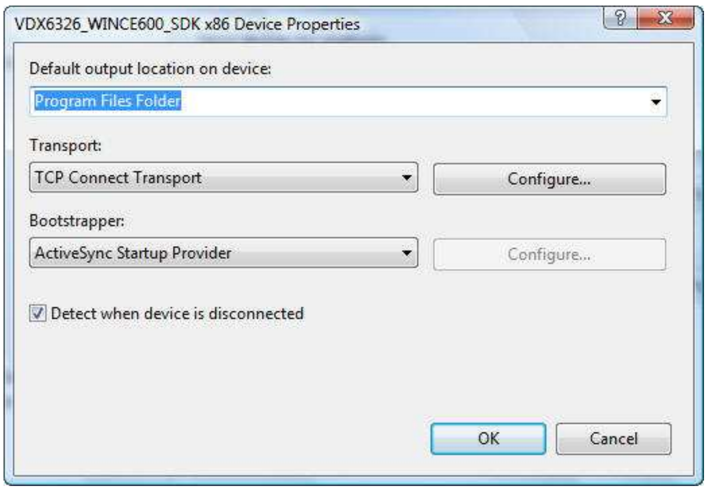
- On the left, click to expand the **Device Tools** node and click on **Devices**.
- On the right, select **VDX6326_WINCE600_SDK** from the list of available platform in the **Show devices for platform** combo text box, as shown in Figure N09.



Show devices for platform: VDX6326_WINCE600_SDK

Fig. N09 - VS2008 Tools Options

- Click on the **Properties** button to bring up **VDX6326_WINCE600_SDK x86 Device Properties** setting screen, as shown in Figure N10.


VDX6326_WINCE600_SDK x86 Device Properties

Default output location on device: Program Files Folder

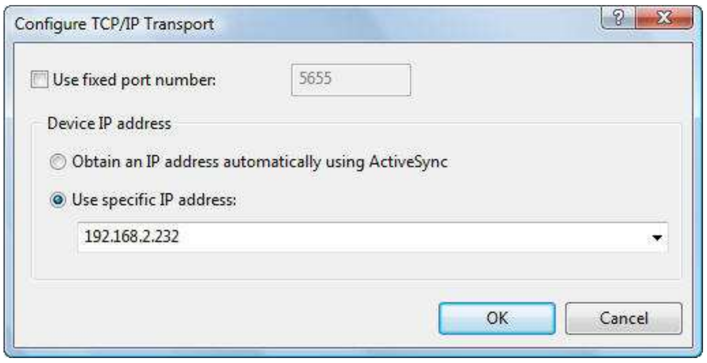
Transport: TCP Connect Transport

Bootstrapper: ActiveSync Startup Provider

[✓] Detect when device is disconnected

Fig. N10 - VDX6326_WINCE600 x86 Device properties

- Click on the **Configure** button to bring up **Configure TCP/IP Transport** screen.
- Select **Use specific IP address** and enter IP address for the target device, as shown in Figure N11.


Configure TCP/IP Transport

[ ] Use fixed port number: 5655

Device IP address

○ Obtain an IP address automatically using ActiveSync

● Use specific IP address: 192.168.2.232

Fig. N11 - Configure TCP/IP Transport / Set device IP Address

- Click the **OK** button to save device IP address setting.
- Click the **OK** button on the **VDX6326_WINCE600_SDK x86 Device Properties** screen.
- Click the **OK** button on the **Options** screen.

Step 3: Connecting Target Device to Visual Studio 2008 IDE using CoreCon

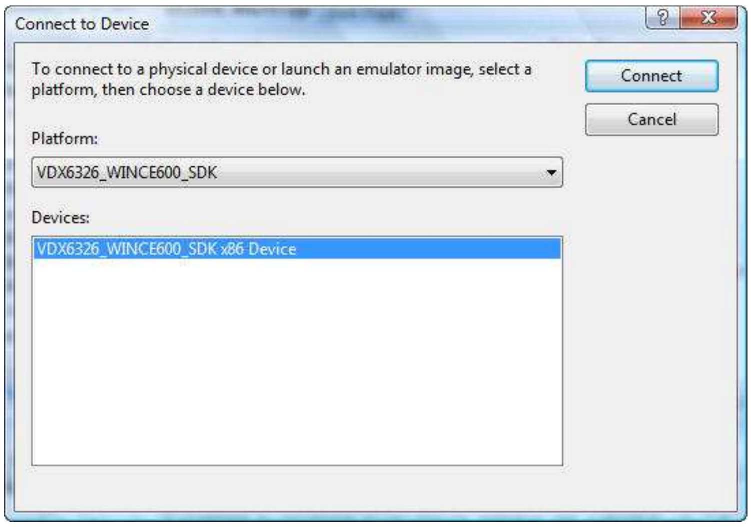
To initiate connection between VS2008 IDE and the target device using CoreCon, we need to launch the **ConmanClient2.exe** and **cMaccept.exe** CoreCon components from the target device manually. This is a cumbersome process and has to perform at least once for each time the target device reset power.

During the OS design customize steps, we added the CoreCon_v200_x86 component, AutoLaunch_v200_x86 component and entered registry entries to launch the CoreCon executable automatically when the CE 6.0 OS starts.

The CoreCon_v200_x86 component is used for development purpose and should not be included in the release image, and should be removed from the image intended for distribution.

With the CoreCon_v200_x86 and AutoLaunch_v200_x86 components included in the runtime image to launch the CoreCon executable when the CE 6.0 OS starts, work through the following steps to connect the VS2008 development workstation to the target device:

- From VS2008 IDE, select **Tools | Connect to device…** and select **VDX6326_WINCE600_SDK** from the list of available devices, and click on the **Connect** button, as shown in Figure N12.


Connect to Device

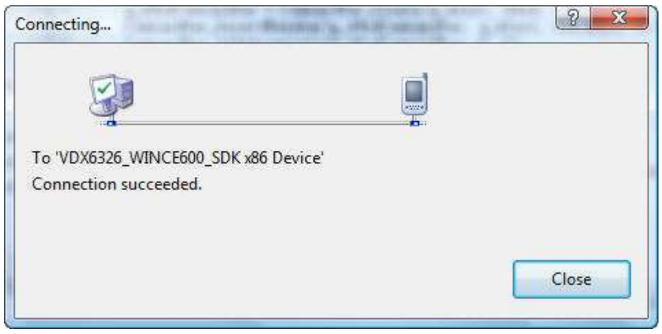
To connect to a physical device or launch an emulator image, select a platform, then choose a device below.

Platform:
VDX6326_WINCE600_SDK

Devices:
VDX6326_WINCE600_SDK x86 Device

Fig. N12 - Connect to Device

When connection is successful, the Connecting dialog box will display Connection succeeded to indicate a successful connection, as shown in Figure N13.


Connecting...

To 'VDX6326_WINCE600_SDK x86 Device'
Connection succeeded.

Fig. N13 - Target device connected

Step 4: Download VS2008_Win32 Application to Target Device

With the CoreCon Connectivity established, we are ready to download the VS2008_Win32 application to the target device.

From VS2008 IDE, select **Debug | Start Debugging** to deploy the **VS2008_Win32** application to the target device.

After the VS2008_Win32 application is downloaded, it will launch on the target device as shown in Figure N14.

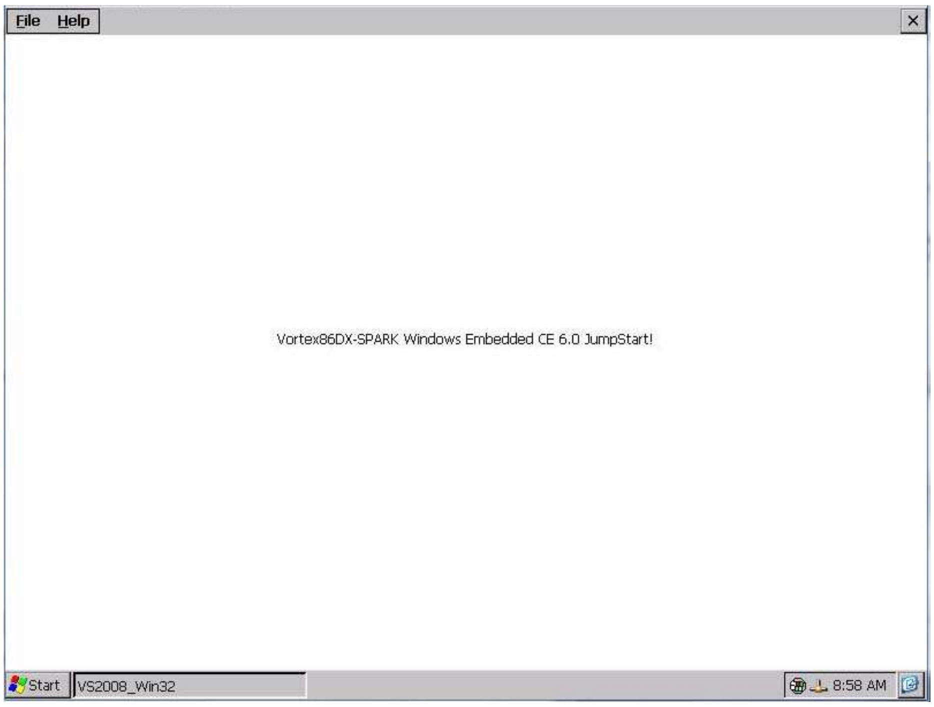


Fig. N14 - VS2008_Win32 application running on the target device

Appendix O – Using VDX-6326 GPIO

There are 16-bits GPIO accessible on the VDX-6326 SBC, through the J11 header.

The 16-bits GPIO are extended from the Vortex86DX SoC's GPIO Port-0 and Port-1, with 8-bits link to each port. Here are the configuration and data register for the GPIO.

	Port 0	**Port 1**	**Description**
Data Register	78H	79H	
Direction Register	98H	99H	0: GPIO pin is input mode 1: GPIO pin is output mode

Using the GPIO

In this section, we will provide sample codes showing how to configure and use the VDX-6326 GPIO.

Windows Embedded CE Sample Codes

Following are two functions, using inline assembly code, to write to and read from the Vortex86DX SoC's register.

```
#include “stdafx.h”

unsigned char inportb(int addr)
{
  __asm
  {
    push edx
    mov edx, DWORD PTR addr
    in al, dx
    and eax, 0xff
    pop edx
  }
}

void outportb(int addr, unsigned char val)
{
  __asm
  {
    push edx
    mov edx, DWORD PTR addr
    mov al, BYTE PTR val
    out dx, al
    pop edx
  }
}
```

Configure GPIO to Output Mode

The GPIO can be used to light up or blink LEDs to function as visual status indicator, to trigger relay or send reset signal to another controller.

Each of the 8-bits on Port-0 and Port-1 can be configured independently to input or output mode, by writing the appropriate value to the direction register at 98H for Port-0 and 99H for Port-1, as follow:

To configure all 8-bits on Port-0 to output mode:

```
Outportb(0x98, 0xff);
```

To configure all 8-bits on Port-1 to output mode:

```
Outportb(0x99, 0xff);
```

To configure only bit-0, bit-1, bit-2 and bit-3 on Port-0 to output mode:

```
Outportb(0x98, 0x0f);
```

To configure only bit-0, bit-1, bit-2 and bit-3 on Port-1 to output mode:

```
Outportb(0x99, 0x0f);
```

To configure bit-4, bit-5, bit-6 and bit-7 on Port-0 to output mode:

```
Outportb(0x98, 0xf0);
```

To configure bit-4, bit-5, bit-6 and bit-7 on Port-1 to output mode:

```
Outportb(0x99, 0xf0);
```

To configure bit-0 and bit-3 on Port-0 to output mode:

```
Outportb(0x98, 0x05);
```

To configure bit-0 and bit-3 on Port-1 to output mode:

```
Outportb(0x99, 0x05);
```

After the GPIO port is configured to output mode, each GPIO bit can be toggle to high or low by writing appropriate value to the data register at 78H for Port-0 and 79H for Port-1, as follow:

To set all 8 GPIO bits on Port-0 to high:

```
Outportb(0x78, 0xff);
```

To set all 8 GPIO bits on Port-1 to high:

```
Outportb(0x79, 0xff);
```

To set only bit-0, bit-1, bit-2 and bit-3 on Port-0 to high:

```
Outportb(0x78, 0x0f);
```

To set only bit-0, bit-1, bit-2 and bit-3 on Port-1 to high:

```
Outportb(0x79, 0x0f);
```

To set only bit-4, bit-5, bit-6 and bit-7 on Port-0 to high:

```
Outportb(0x78, 0xf0);
```

To set only bit-4, bit-5, bit-6 and bit-7 on Port-1 to high:

```
Outportb(0x79, 0xf0);
```

To set only bit-0 and bit-3 on Port-0 to high:

```
Outportb(0x78, 0x05);
```

To set only bit-0 and bit-3 on Port-1 to high:

```
Outportb(0x79, 0x05);
```

To set all 8 GPIO bits on Port-0 to low:

```
Outportb(0x78, 0x00);
```

To set all 8 GPIO bits on Port-1 to low:

```
Outportb(0x79, 0x00);
```

Note: When a GPIO bit is set to high, it provides 3.3V and able to supply up to 16 mA of current, sufficient current to drive an LED.

When connecting an LED to the GPIO, make sure to add appropriate resistor to minimize current flow to prevent damage to the SBC.

Configure GPIO to Intput Mode

When configured to input mode, the GPIO can be used to detect the status of a push button, relay, or another device with appropriate electronic circuitry.

Each of the 8-bits on Port-0 and Port-1 can be configured independently to input or output mode, by writing the appropriate value to the direction register at 98H for Port-0 and 99H for Port-1, as follow:

To configure all 8-bits on Port-0 to output mode:

```
Outportb(0x98, 0x00);
```

To configure all 8-bits on Port-1 to output mode:

```
Outportb(0x99, 0x00);
```

Note: There is one configuration bit to configure the GPIO bit to either input or output mode. If the GPIO bit is not configured to “input” mode, it’s in “output” mode.

When configured to output mode, the GPIO bit measures 2.45V.

After the GPIO bit is configured to input mode, it measures 2.45V with associated data bit set to “1”.

When a GPIO bit, configured to input mode, is grounded, the associated data bit is changed to “0”. After the GPIO bit is disconnected from the grounding signal, the associated data bit changes back to “1”.

To capture the status of a GPIO bit configured to input mode, you need write a routine to poll the associated data bit for changes.

To read data associated with GPIO bits configured to input mode, use the inportb() function to read data from 78H for Port-0 and 79H for Port-1, as follow:

To read data on Port-0:

```
inportb(0x78);
```

To read data on Port-1:

```
inportb(0x79);
```

GPIO.DLL

The GPIO.DLL is provided to help managed code developer to access and use the GPIO. The GPIO.DLL is provided as part of the BSP and include in resulting CE 6.0 OS runtime image when this BSP is in used to develop the OS design.

The GPIO.DLL exposes the following API to configure GPIO input/output mode, set individual GPIO bit high or low and read GPIO data:

- SetPort0Dir(byte dbDirection)
  Function to set GPIO Port-0 to input or output mode

- SetPort1Dir(byte dbDirection)
  Function to set GPIO Port-1 to input or output mode

- SetPort2Dir(byte dbDirection)
  Function to set GPIO Port-2 to input or output mode

- SetPort3Dir(byte dbDirection)
  Function to set GPIO Port-3 to input or output mode

- GetPort0Dir()
  Function to retrieve setting for GPIO Port-0

- GetPort1Dir()
  Function to retrieve setting for GPIO Port-1

- GetPort2Dir()
  Function to retrieve setting for GPIO Port-2

- GetPort3Dir()
  Function to retrieve setting for GPIO Port-3

- ReadPort0()
  Function to read data register for GPIO Port-0

- ReadPort1()
  Function to read data register for GPIO Port-1

- ReadPort2()
  Function to read data register for GPIO Port-2

- ReadPort3()
  Function to read data register for GPIO Port-3

- WritePort0(byte dbValue)
  Function to write to the data register for GPIO Port-0

- WritePort1(byte dbValue)
  Function to write to the data register for GPIO Port-1

- WritePort2(byte dbValue)
  Function to write to the data register for GPIO Port-2

- WritePort3(byte dbValue)
  Function to write to the data register for GPIO Port-3

Sample codes showing how to use the GPIO.DLL is provided on the jump start CD, under the `\SampleCodes\GPIO\` folder.

Appendix P – Using VDX-6326 Watchdog Timers

There are 2 watchdog timers accessible on the VDX-6326 SBC, WDT0 and WDT1.

The codes used to set, reset and disable WDT0 are different from WDT1.

Using WDT0

To access WDT0 registers, use index port 22H and data port 23H. WDT0 uses 32.768 KHz frequency source to count a 24-bit counter, and able to support timer interval ranging from 30.5 µ seconds to 512 seconds. WDT0 can be configured to trigger system reset, NMI or IRQ.

Sample codes to configure and use WDT0:

```
void main(void)
{
  unsigned char c;
  unsigned int lTime;

  outp(0x22,0x13); // Lock register
  outp(0x23,0xc5); // Unlock config. register

  // Configure WDT0 to trigger in 5 seconds
  lTime = 0x20L * 5000L;
  outp(0x22,0x3b);
  outp(0x23,(lTime>>16)&0xff);
  outp(0x22,0x3a);
  outp(0x23,(lTime>> 8)&0xff);
  outp(0x22,0x39);
  outp(0x23,(lTime>> 0)&0xff);

  // Configure WDT0 to reset the system when triggered
  outp(0x22,0x38);
  c = inp(0x23);
  c &= 0x0f;
  c |= 0xd0; // 0xd0 configure the WDT0 to reset the system
             // Change to 0x10 to configure WDT0 to trigger IRQ3
             // Change to 0x20 to configure WDT0 to trigger IRQ4
             // Change to 0x30 to configure WDT0 to trigger IRQ5
             // Change to 0x40 to configure WDT0 to trigger IRQ6
             // Change to 0x50 to configure WDT0 to trigger IRQ7
             // Change to 0x60 to configure WDT0 to trigger IRQ9
             // Change to 0x70 to configure WDT0 to trigger IRQ10
             // Change to 0x90 to configure WDT0 to trigger IRQ12
             // Change to 0xa0 to configure WDT0 to trigger IRQ14
             // Change to 0xb0 to configure WDT0 to trigger IRQ15
             // Change to 0xc0 to configure WDT0 to trigger NMI
             // Change to 0xd0 to configure WDT0 to trigger system reset

  outp(0x22,0x38);
  outp(0x23,c);

  // Enable watchdog timer
  outp(0x22,0x37);
  c = inp(0x23);
  c |= 0x40;
  outp(0x22,0x37);
  outp(0x23,c);

  outp(0x22,0x13); // Lock register
  outp(0x23,0x00); // Lock config. register
  printf("Press any key to stop trigger timer.\n");
  while(!kbhit())
  {
    outp(0x22,0x13);  // Unlock register
    outp(0x23,0xc5);
    outp(0x22,0x3c);
    unsigned char c = inp(0x23);
    outp(0x22,0x3c);
    outp(0x23,c|0x40);
    outp(0x22,0x13);  // Lock register
    outp(0x23,0x00);
  }

  printf("System will reboot after 5 seconds.\n");
}
```

Using WDT1

WDT1 does not use index and data port to access the watchdog timer registers. It uses I/O port 68H ~ 6DH. WDT1 can be configured to trigger system reset, NMI or IRQ.

Sample codes to configure and use WDT1:

```
void main(void)
{
  unsigned char c;
  unsigned long lTime;

  // Configure WDT1 to trigger in 5 seconds
  lTime = 0x20L * 5000L;
  outp(0x6c, (lTime >> 16) & 0xff);
  outp(0x6b, (lTime >>  8) & 0xff);
  outp(0x6a, (lTime >>  0) & 0xff);

  // Configure WDT1 to reset system when it trigger
  outp(0x69, 0xd0);  // outp(0x69, 0x10) to trigger IRQ3
                     // outp(0x69, 0x20) to trigger IRQ4
                     // outp(0x69, 0x30) to trigger IRQ5
                     // outp(0x69, 0x40) to trigger IRQ6
                     // outp(0x69, 0x50) to trigger IRQ7
                     // outp(0x69, 0x60) to trigger IRQ9
                     // outp(0x69, 0x70) to trigger IRQ10
                     // outp(0x69, 0x90) to trigger IRQ12
                     // outp(0x69, 0xa0) to trigger IRQ14
                     // outp(0x69, 0xb0) to trigger IRQ15
                     // outp(0x69, 0xc0) to trigger NMI
                     // outp(0x69, 0xd0) to trigger system reset

  // Enable WDT1 watchdog timer
  c = inp(0x68);
  c |= 0x40;
  outp(0x68, c);

  printf("Press any key to stop trigger timer.\n");
  while(!kbhit())
    outp(0x67, 0x00);

  printf("System will reboot after 5 seconds.\n");

}
```

WDT.DLL

The WDT.DLL is provided as part of the BSP and include in resulting CE 6.0 OS runtime image when this BSP is in used to develop the OS design.

There are six functions in the WDT.DLL:

```
// Set watchdog timer
int  SetWDT0(unsigned int nTime, unsigned char nEvent);
int  SetWDT1(unsigned int nTime, unsigned char nEvent);

// Reset watchdog timer
void ResetWDT0(void);
void ResetWDT1(void);

// Disable watchdog timer
void DisableWDT0(void);
void DisableWDT1(void);
```

Sample managed code project showing how to use the WDT.DLL is provided on the jump start CD, under the `\SampleCodes\WatchdogTimer\` folder.

Appendix Q – Hyper Terminal

While HyperTerminal has been part of the Windows XP OS, it has been removed from the Windows Vista OS.

The HyperTerminal application from a Windows XP workstation can be used on a Windows Vista workstation.

If you still have a Windows XP machine around, locate and copy the following two HyperTerminal application files from the Windows XP machine and place these two files to the same folder on the Windows Vista machine:

- \Program Files\Windows NT\Hypertrm.exe
- \Windows\System32\Hypertrm.dll

On the Windows Vista machine, launch the Hypertrm.exe executable to start HyperTerminal.