i.MX31 SOM-LV Windows Embedded CE 6.0
Lab 1: Intro
BSP Documentation

Logic // Products
Published: June 2009

Revision History

REV	EDITOR	DESCRIPTION	APPROVAL	DATE
A	MH	Initial release	JCA	06/09/09

Table of Contents

1 Introduction

1.1 Learning Objectives

- Create an OS Design using Visual Studio
- Identify the catalog features included in the design
- Extend the standard design by adding catalog items
- Setup the build configuration for the run-time image and build a run-time image
- Incorporate files and applications in your image
- Download the image to the board
- Create, incorporate, and test a Device Driver

1.2 Prerequisites

Knowledge of the vocabulary used in OS design, Visual Studio, and the Platform Builder Plug-in for Visual Studio

1.3 Lab Setup

To complete this lab, you must have:

- A development workstation running Windows XP or Vista
- Visual Studio 2005 (Version 8)
- Service Pack 1 for Visual Studio 2005
- Service Pack 1 for Visual Studio 2005 on Vista (if you are running Vista)
- Windows CE 6.0 R2 Platform Builder plug-in with Service Pack 1
- A Zoom i.MX31 LITEKIT
- The i.MX31 SOM-LV Windows CE 6.0 BSP

2 Create the Platform Image

During the first part of this lab you will use the New OS Design Wizard within Platform Builder to create an initial OS Design which you will then modify by adding applications and modifying the registry.

- Start Visual Studio 2005 with Platform Builder either using the desktop shortcut, or…
- **Start** | **All Programs** | **Microsoft Visual Studio 2005** | **Microsoft Visual Studio 2005** (there may also be a shortcut on the desktop)

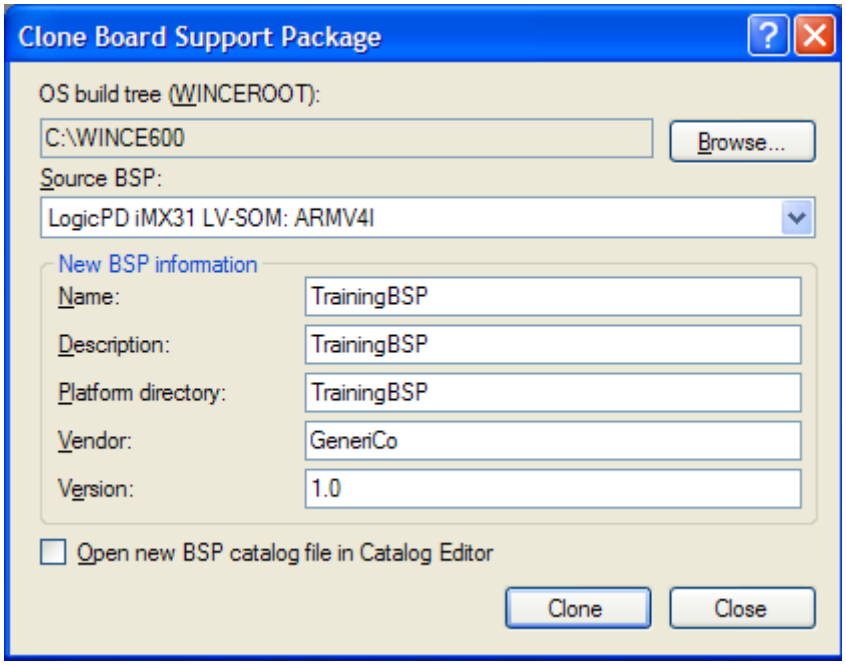
In this exercise, you will use the Clone BSP tool in Visual Studio 2005 to create a copy of the existing Hardware Board Support Package (BSP). We can modify this copy instead of modifying the original that was delivered with your i.MX31 LITEKIT.

2.1 Clone the BSP

1. Launch **Microsoft Visual Studio 2005**.

Note: If this is the first time Visual Studio is launched after installation the “Choose Default Environment Setting” dialog will be displayed. For the purposed of this course select Platform Builder Development Settings and select Start Visual Studio.

2. Select the **Tools** | **Platform Builder for CE6.0** | **Clone BSP** from the menu in Visual Studio to bring up the Clone BSP dialog box.
3. In the Clone BSP dialog select **LogicPD iMX31 LV-SOM: ARMV4I** from the Source BSP: drop down box.
4. Type **TrainingBSP** in the Name field in the New Board Support Package Info area.
5. Type a description for your new BSP in the Description field.
6. Type **TrainingBSP** in the Platform Directory field.
7. Type **GeneriCo** in the Vendor field.
8. Type **1.0** in the Version field.



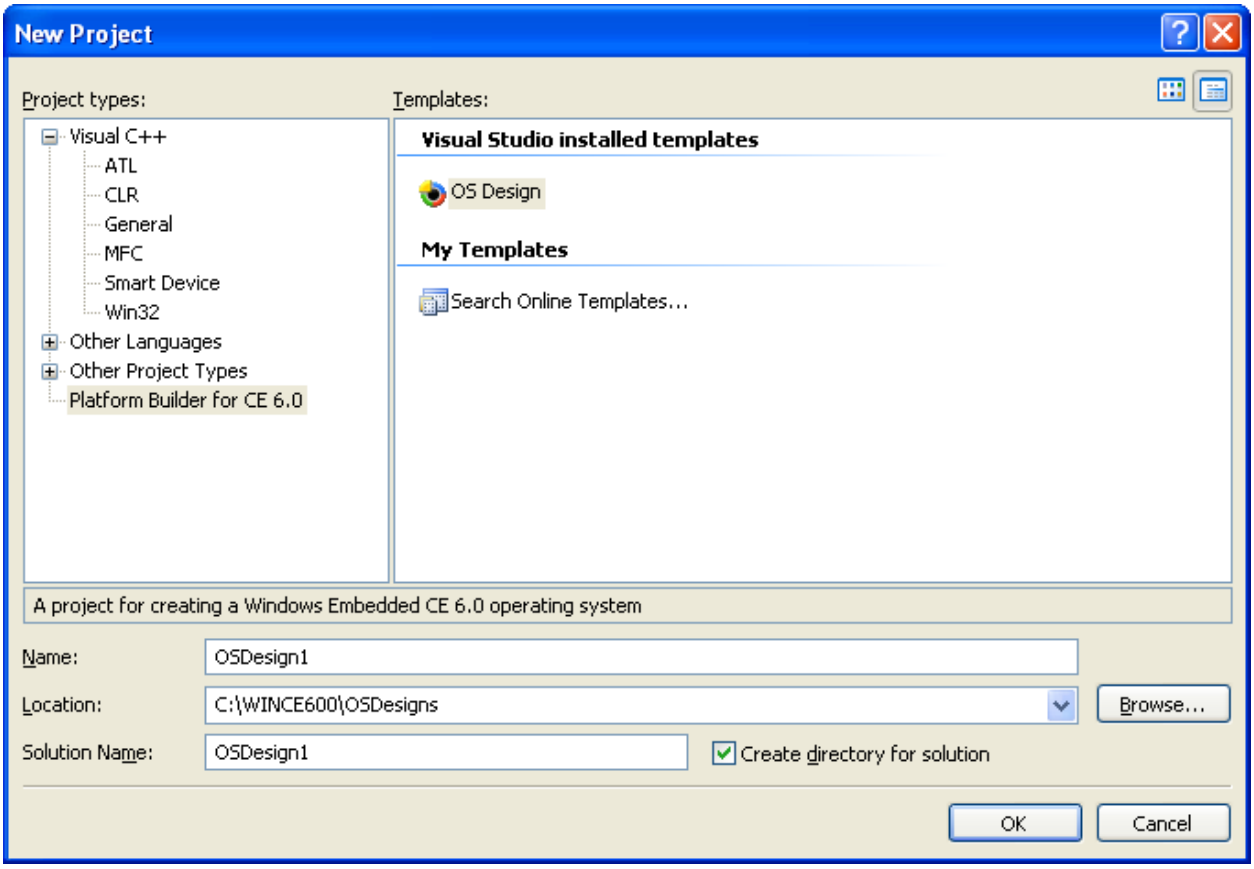
9. Click the **Clone** Button. The Clone BSP tool will create a new Board Support Package based on the Hardware Board Support Package.
10. Acknowledge the **Clone BSP** success message by clicking OK.
11. Open the **C:\WINCE600\PLATFORM\TrainingBSP** folder using Windows Explorer.

The BSP has now been cloned into a new Board Support Package called **TrainingBSP**. The **TrainingBSP Board Support Package** will be used in the remaining labs.

2.2 Create a New OS Design

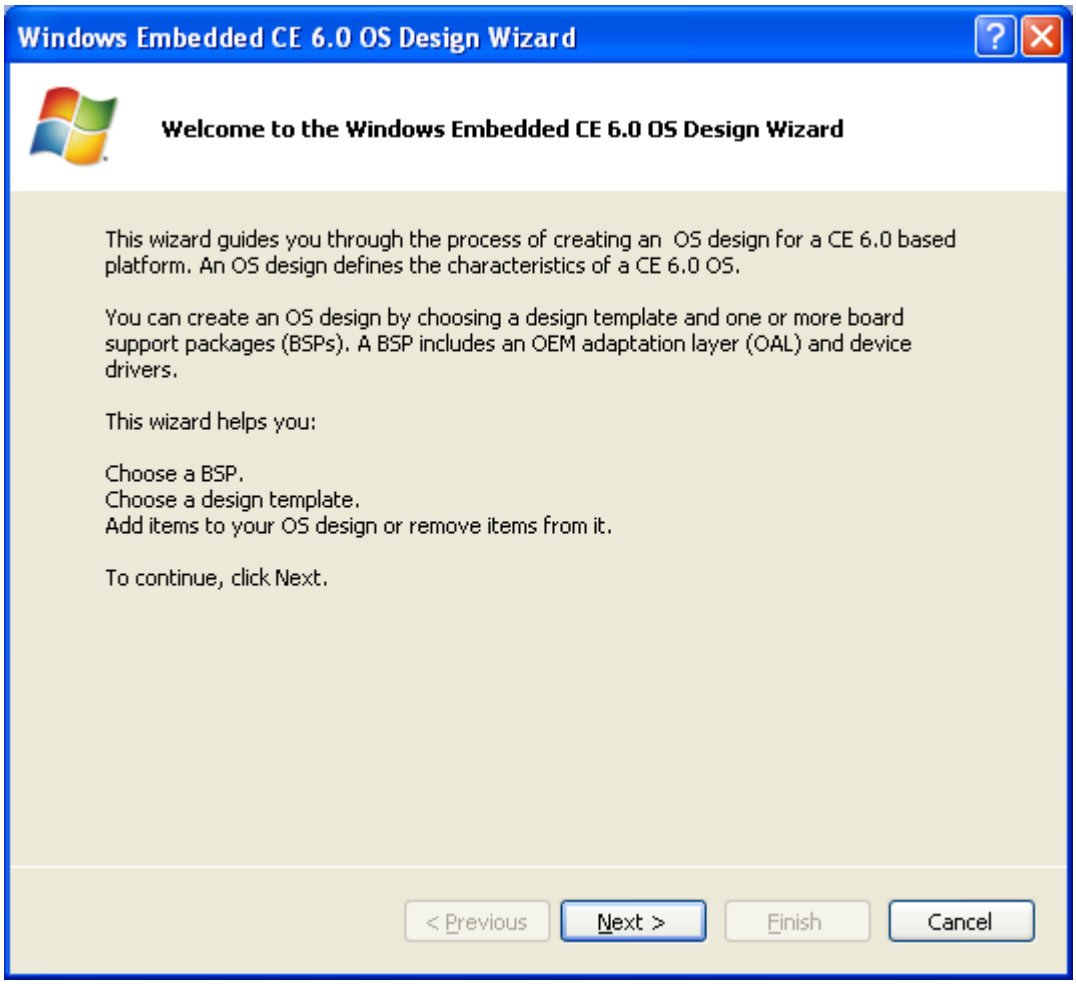
Launch the Visual Studio 2005 New Project dialog.

- Select **File** | **New…** | **Project…**



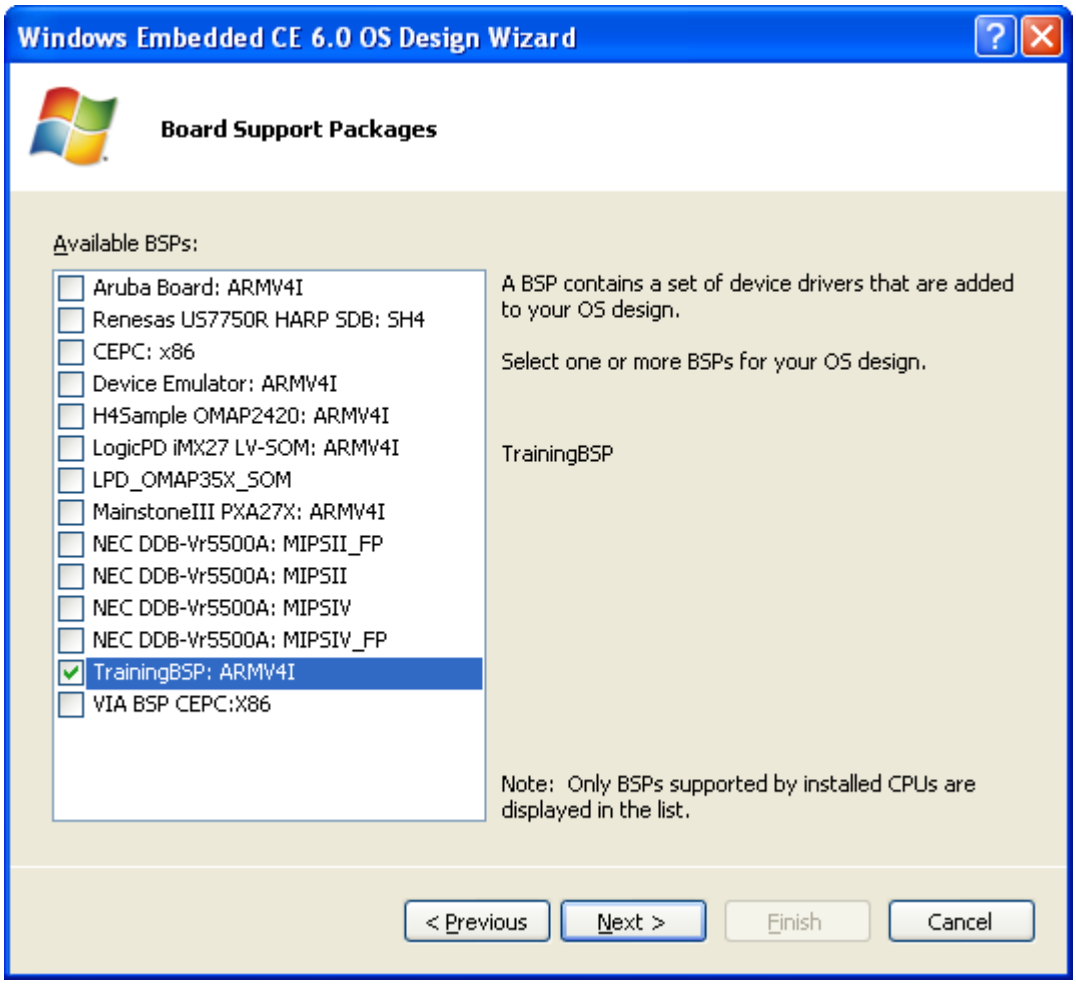
- Select the **Platform Builder for CE 6.0** project type.
- Choose the **OS Design** template
- Enter a name for your project, or go with the default name
- Click **OK**

The initial Dialog that is displayed below outlines the process of creating an OS Design. There are a number of steps involved and you will step through the wizard and make selections as appropriate.



We will build an operating system for the i.MX31 SOM-LV device.

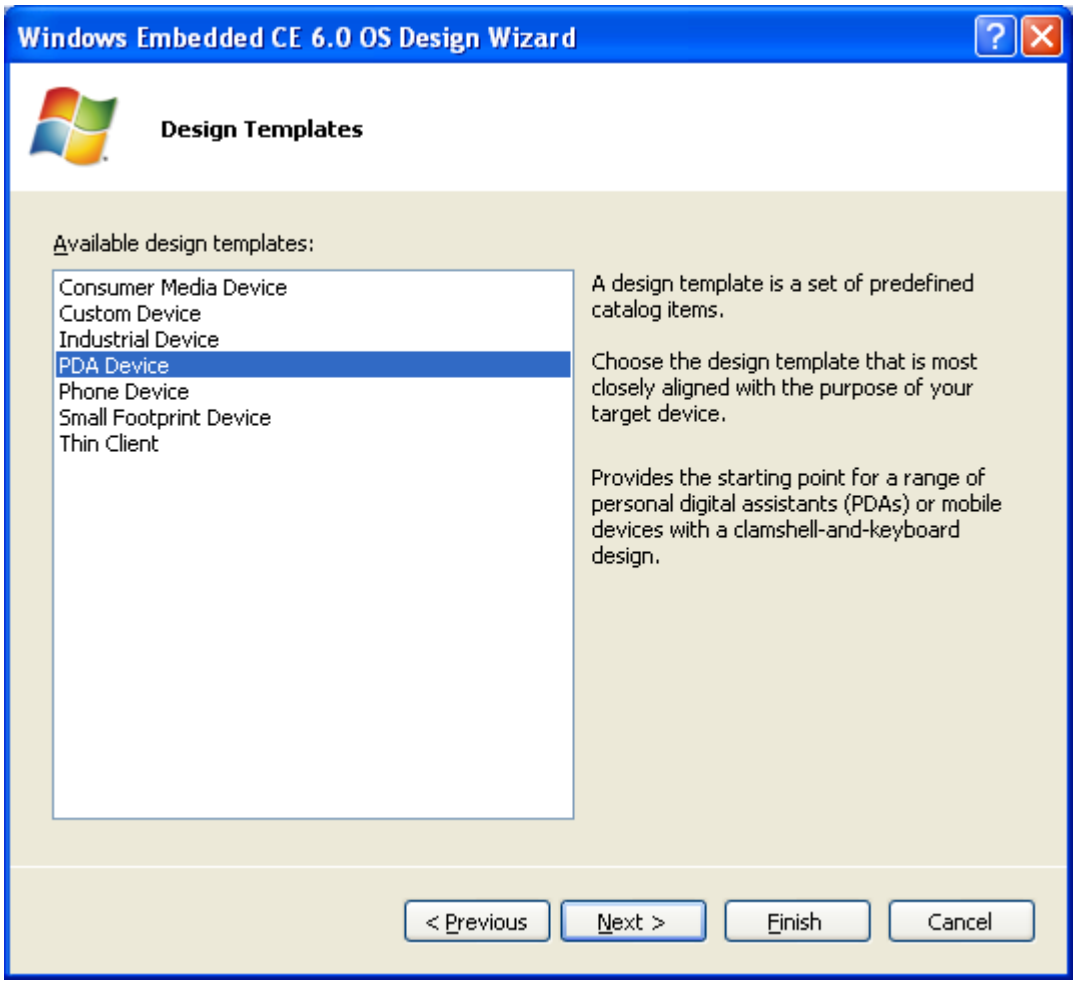
You could select more than one BSP (Board Support Package). For example, it may be useful to select the Device Emulator and a hardware reference platform; the emulation environment can be used by your internal/external application developers to build applications for your device while hardware is still being developed.



- Select **TrainingBSP: ARMV4I**
- Click **Next**

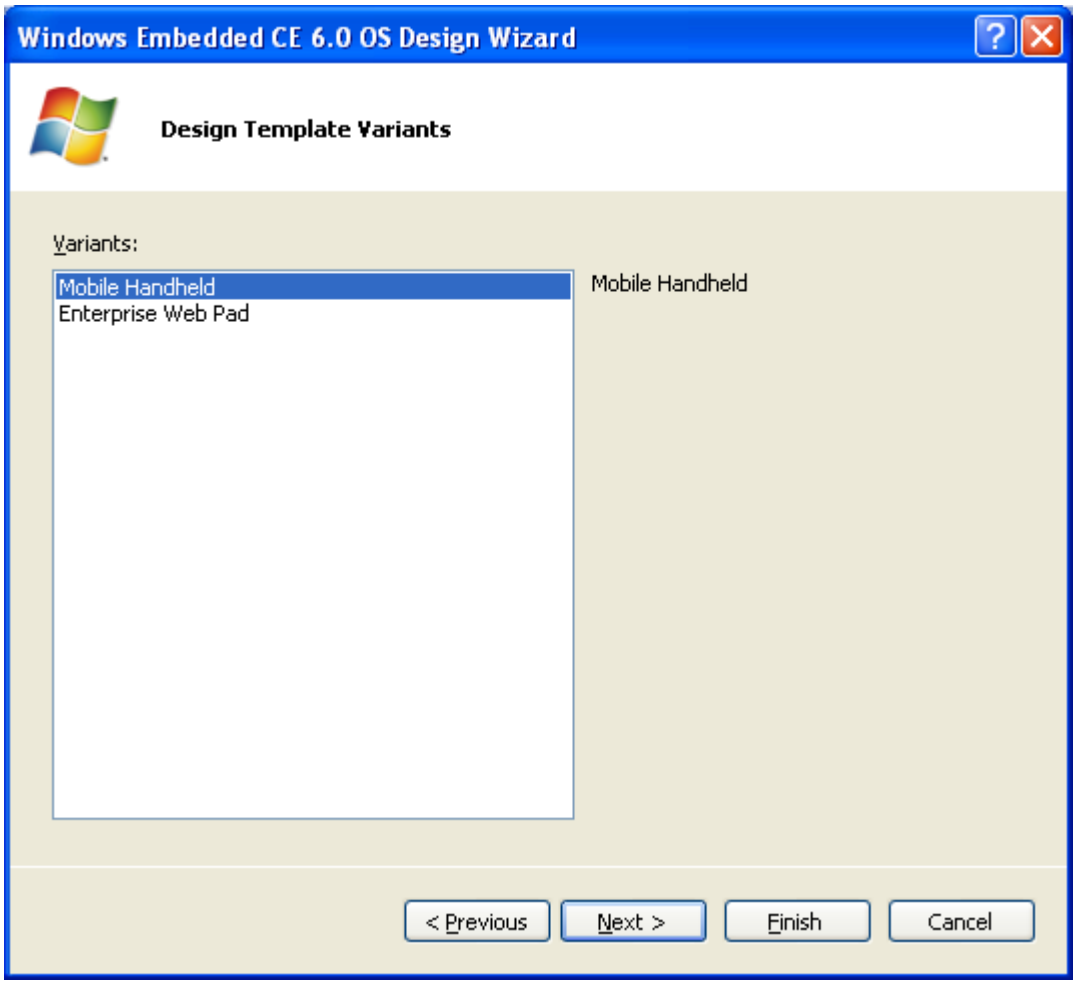
You are now given a number of Design Templates to choose from as the starting point for this project. If none of the options match your needs you can simply select **'Custom Device'** and build the image based on the components you select from the catalog.

For the purposes of this tutorial you will be using the **PDA Device** template.



- Select **PDA Device** from the list of Design Templates
- Click **Next**

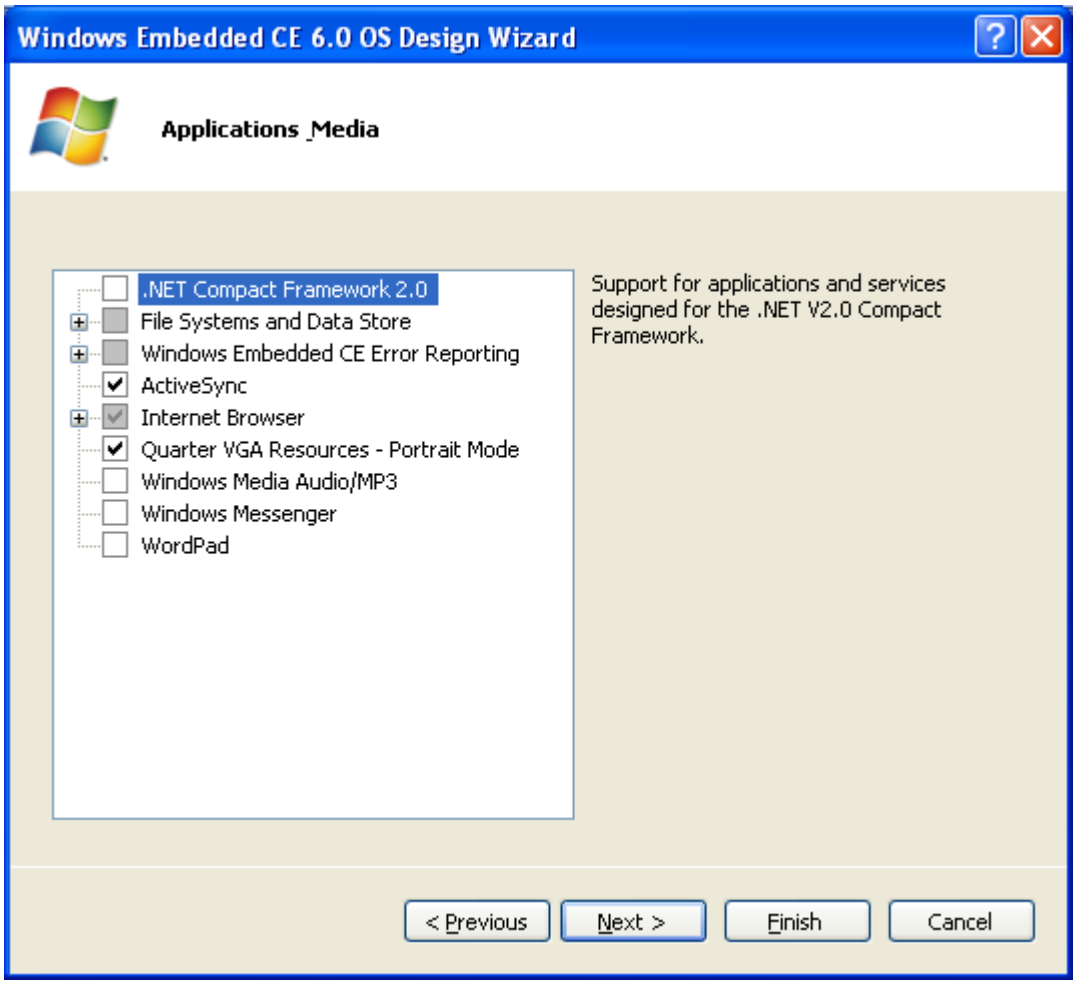
The Wizard now gives you the option of choosing a variant of the Design Template. For this tutorial you will be using the Mobile Handheld variant.



- Select **Mobile Handheld** from the list of Design Template Variants
- Click **Next**

There are a number of options which can be selected for each of the sample platforms, the OS Design Wizard will only show options that are relevant to the platform you are building. For example, it would not make sense to include Internet Explorer or WordPad in a headless device such as a Gateway. The Mobile Handheld device can include applications such as Internet Explorer, WordPad, and Windows Messenger.

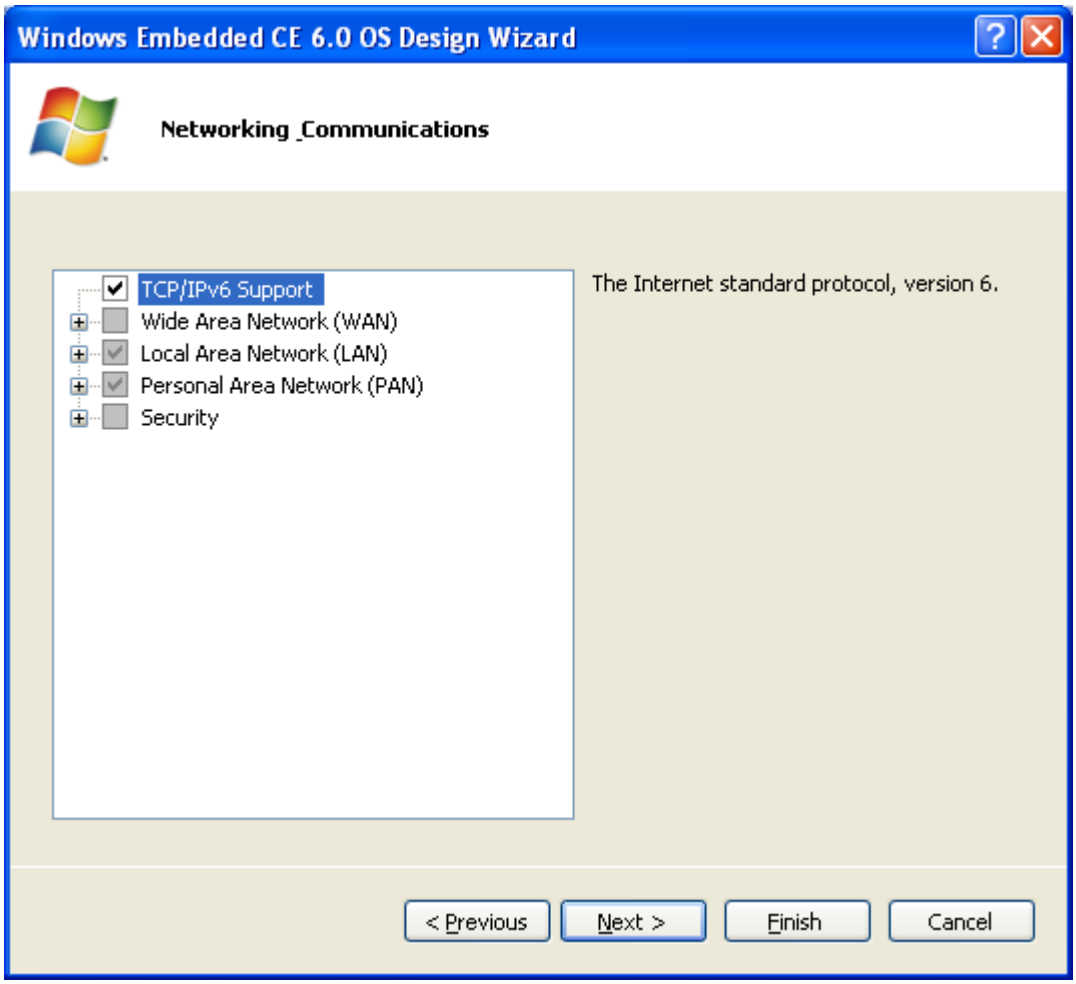
In this case we don't need the .NET Compact Framework. De-select this item.



- De-select the **.NET Compact Framework**
- Click **Next**

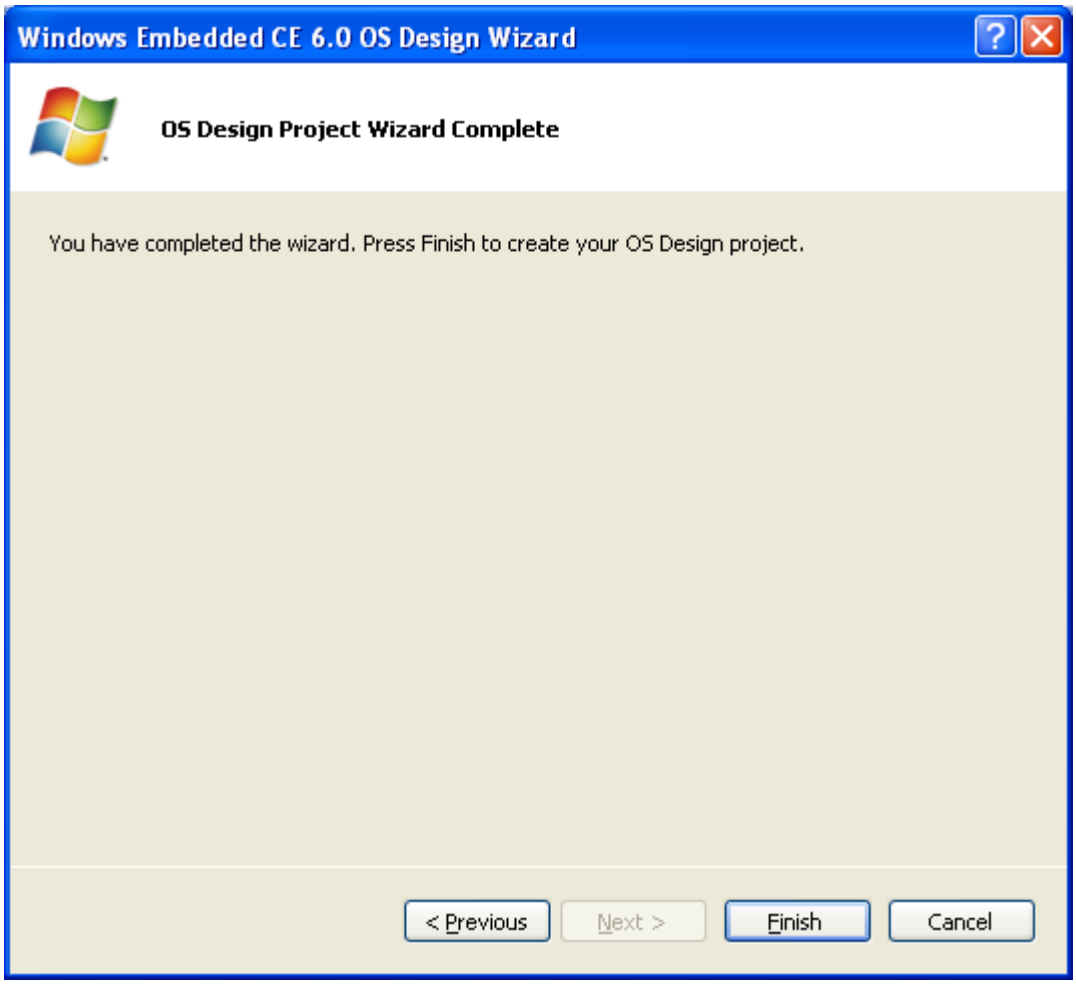
Next comes networking and communications, you can see that Windows CE provides support for Personal, local, and wide area networking through technologies such as Bluetooth, IrDA, Wired and Wireless networking, and VPN.

In our case the default options are fine.



- Click **Next**

We're done! – The wizard is complete. You've configured the OS Design and can now further customize it by adding and/or removing components from the Catalog.



- Click **Finish**

The Wizard will now prompt you with any security warnings or other notifications related to the components you selected. Acknowledge these notifications.

- Click **Acknowledge**

At this point you have an OS Design containing all of the OS components selected from the Wizard. In the next section we will further customize the OS Design using the Catalog.

3 Customize and Build the OS Design

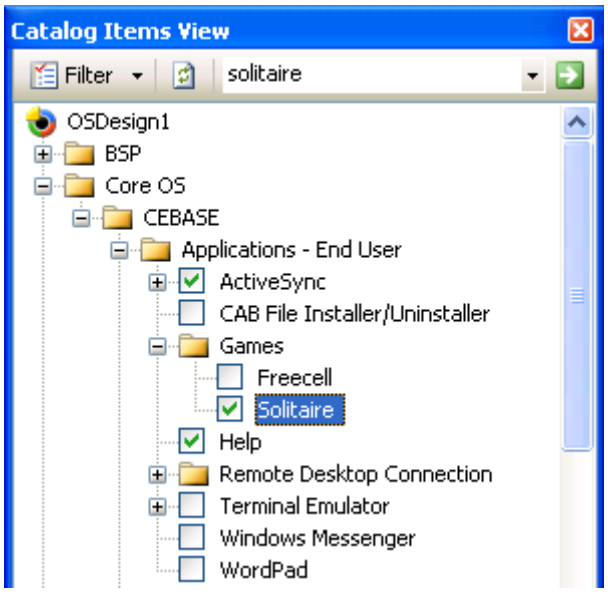
3.1 Using the Catalog

You will now make changes to the items included in your OS Design using the Catalog View.

First, ensure that the Catalog View is visible.

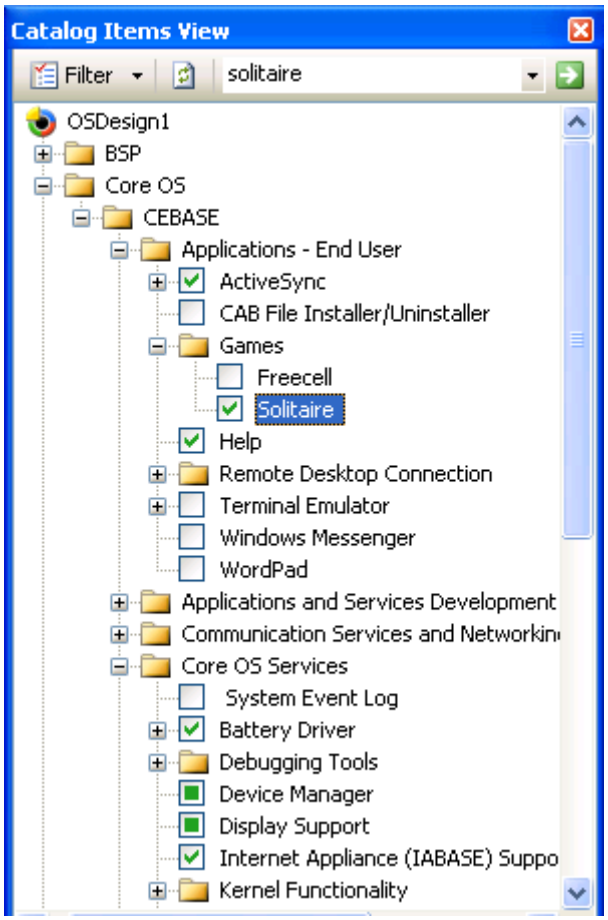
- Select the **View** menu
- Choose **Other Windows** | **Catalog Items View**

You want to ensure that Solitaire is included in the run-time image that you build. Search for “Solitaire” to locate the appropriate catalog item.



- Type **Solitaire** in the search box
- Press the **Enter** key to search
- **Check** the box next to the **Solitaire** item to include it in your OS Design

By default, the Catalog View displays all items that are available to include in an OS Design. Sometimes it is helpful to see only those items that are already included. This is done using the Filter selector.



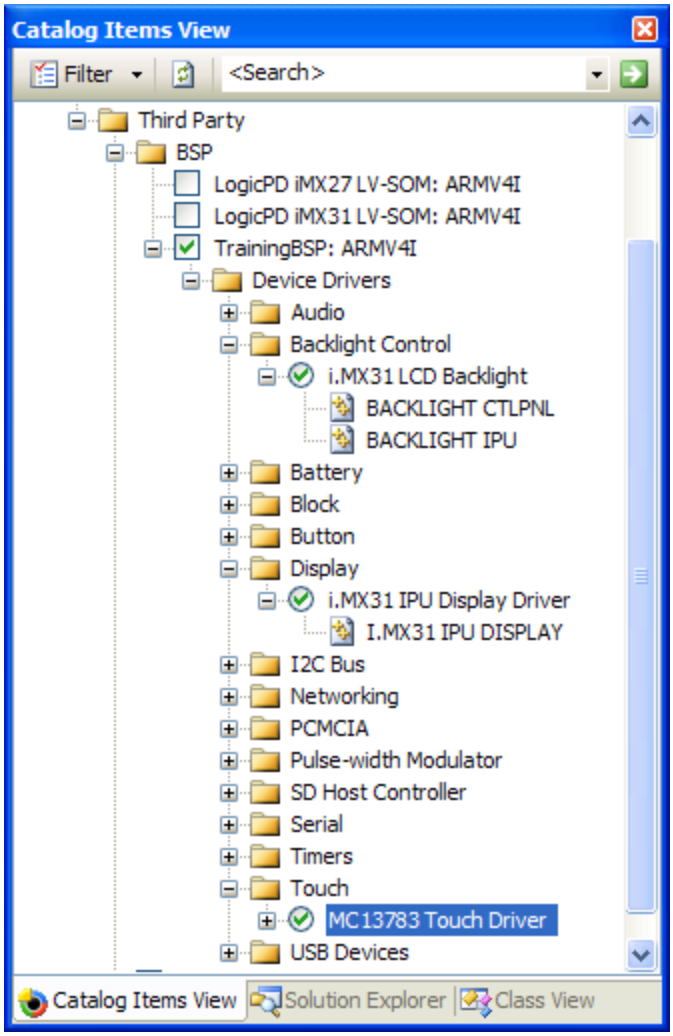
- Select the Filter item from the Catalog Items View Toolbar
- Choose **User-Selected Catalog Items and Dependencies**

If you browse through the catalog in this view, you will notice that Catalog Items that were manually selected or chosen by the OS Design wizard are displayed with a ☑ icon. Those that were brought in automatically as dependencies are shown with a ■ icon.

- Return to the original view by selecting **All Catalog Items in Catalog** from the filter menu

We need to select a few other components for our i.MX31 SOM-LV device.

- In the search box, type **i.MX31 LCD Backlight**
- Select the **i.MX31 LCD Backlight** component
- Note that there are other device drivers that are a part of this BSP.
- Under **Display**, select **i.MX31 IPU Display Driver**
- Under **Touch**, select **MC13783 Touch Driver**
- Your selections should match the following image:



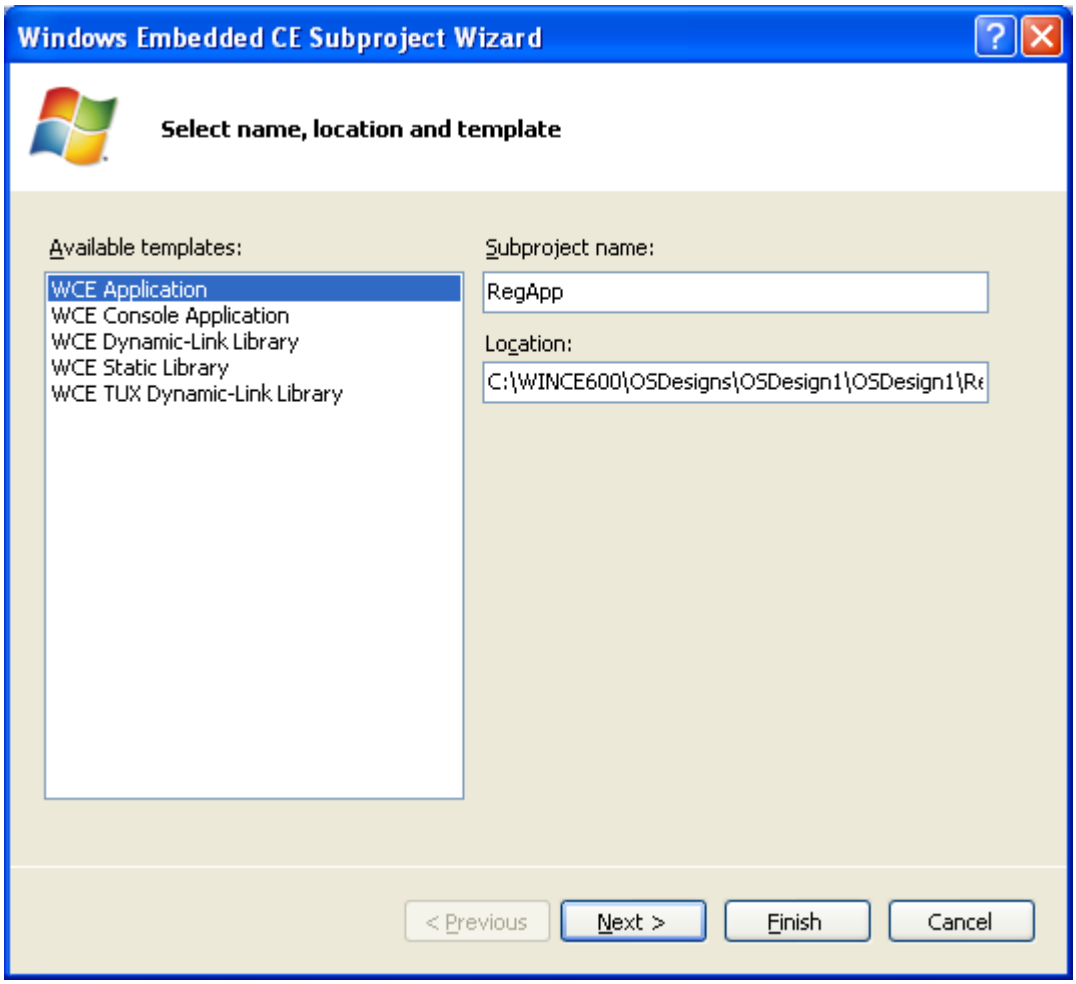
The Catalog is an important tool for configuring your OS Design. If you choose to explore this view further, you will find advanced features for the analysis of dependencies between the various items.

3.2 Adding an Application to the OS Design

The application we'll be adding is a Windows application with UI. We will add a custom registry key to the platform which will be read by the application at startup. If the key is successfully added to the platform we will display the contents of the key to the user in a MessageBox.

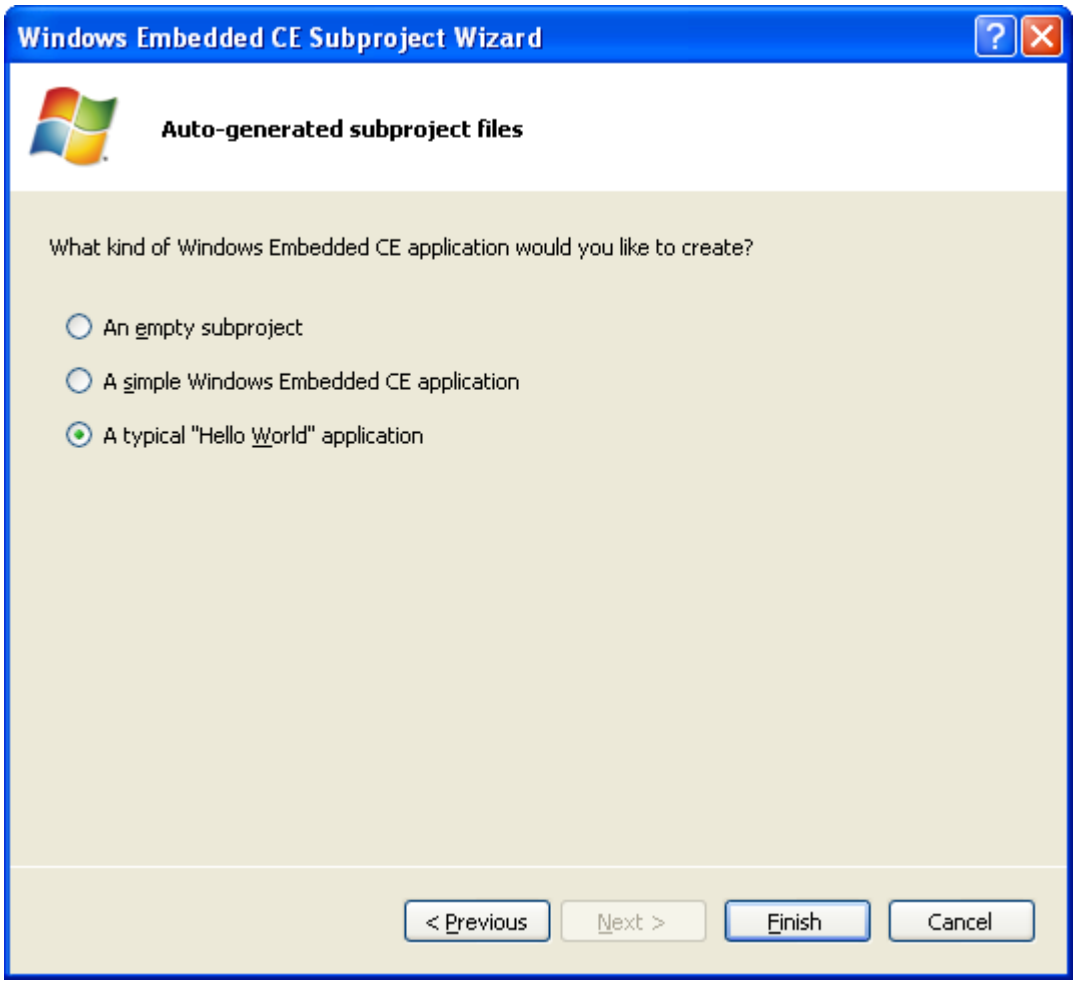
- Switch from the Catalog View to the **Solution Explorer** view
- In the Solution Explorer, right click on the **Subprojects** folder.
- Select **Add New Subproject…**

The Subproject Wizard is displayed and shows a number of different subproject types. Each of the available types is described in the Platform Builder online help.



- Choose **WCE Application**
- Type the name of your application – **RegApp**
- Click **Next**

You will be prompted for the type of application you want to add. There are three types of application. An empty project assumes that you will add all the source files needed for the project. A “Simple” Windows CE application will create the core source files needed to build the project and will have an entry point of WinMain. Finally, a typical “Hello World” application will output the string “Hello World” to the screen when executed.

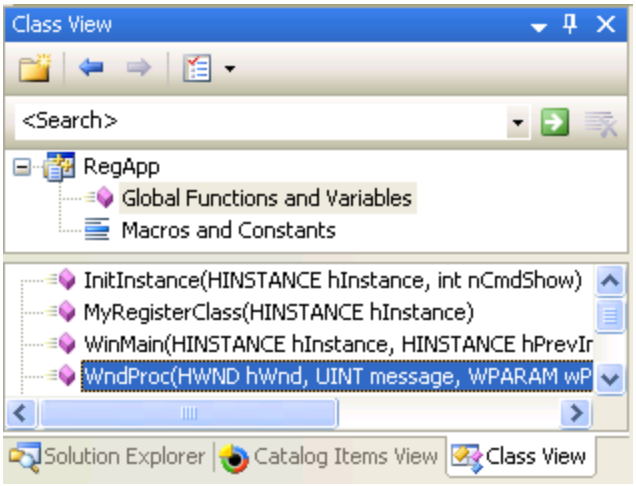


- Select **A typical “Hello World” application**
- Click **Finish**

At this point you have a fully featured “Hello World” application. You could build and deploy the application with no further changes to the code; however, you will want to modify the application to include some code to read a registry key and display the results to the user.

- Switch to **Class View** tab below the Solution Explorer.
- Expand **RegApp** Class
- Select **Global Functions and Variables**

This will expose the functions within the RegApp application. WndProc is the function that deals with Windows Messages.

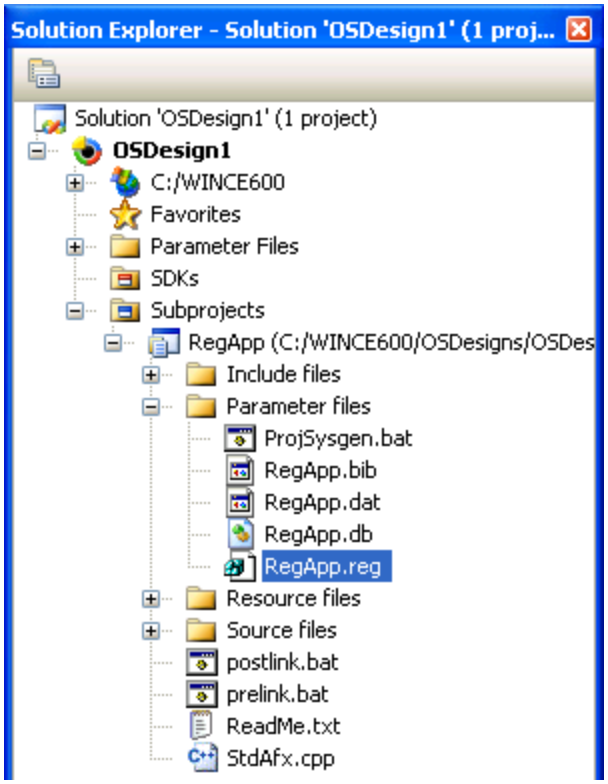


- Double click on the **WndProc** function to open the source for this function.
- **Locate the switch** (message) which is just before the case WM_PAINT within the Window procedure.
- Place your cursor immediately after the open curly brace.
- Add the Registry application code to the application by opening **C:\Labs\Lab1\LabFiles\RegApp.cpp**
- Select the contents of the file and paste them into the code window after the open curly brace you found previously

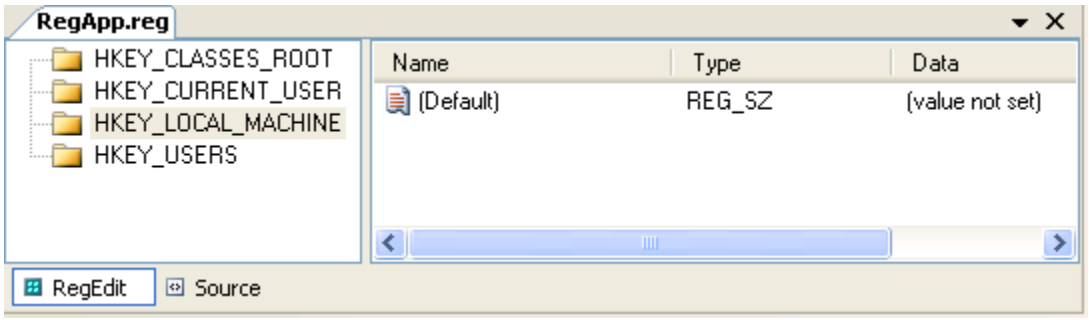
3.3 Adding Registry information to the Platform

The code which you have just added to the RegApp application is used to read a custom registry key on your Windows CE platform – You will now add this key to the platform.

- Switch to the **Solution Explorer**
- Expand the Subproject Folder to reveal **Subprojects** | **RegApp** | **Parameter files** | **RegApp.reg**



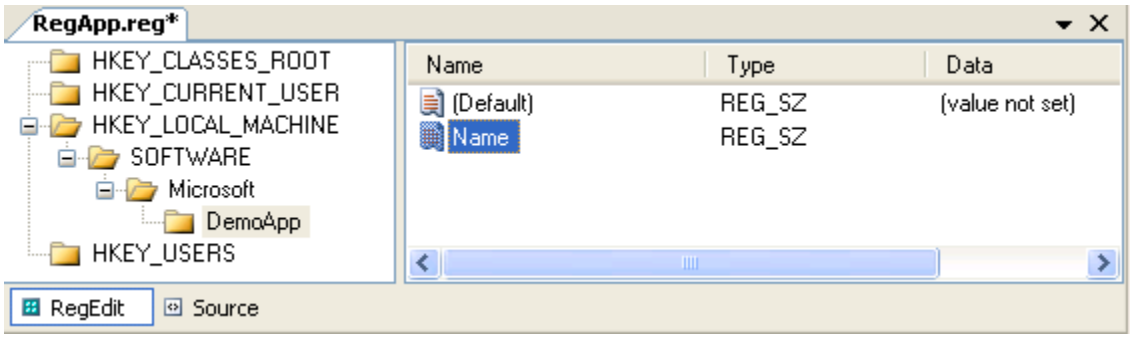
- Double click **RegApp.reg** to open this file for editing



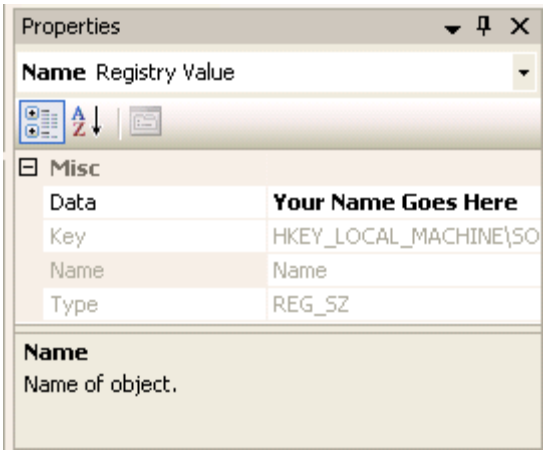
The Reg file is opened with the new graphical “RegEdit” view. You will add the following key to your Reg file:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\DemoApp]
```

- Right-click “HKEY_LOCAL_MACHINE” and select **New** | **Key**
- Name the new key **SOFTWARE**
- Right-click “SOFTWARE” and select **New** | **Key**
- Name the new key **Microsoft**
- Right-click “Microsoft” and select **New** | **Key**
- Name the new key **DemoApp**
- Right-click “DemoApp” and select **New** | **String Value**
- Name the new string value **Name**



- Select **Name**
- Press **Alt+Enter** to bring up the Properties pane
- In the Properties pane, **enter your name** in the Data field



- **File** | **Save RegApp.reg** to save the registry changes
- Switch back to the **Solution Explorer**

3.4 Include a File

In this section we will look at how you include a custom file into the Operating System. This is accomplished by modifying a BIB (Binary Image Builder) file. Your goal will be to include the "Greenstone.bmp" file in the OS image.

- Expand the **RegApp** Subproject in the Solution Explorer
- Expand the **Parameter Files** folder
- Double click **RegApp.bib**
- Add the **FILES** line and the **line below** to ensure that the bitmap is included in the image. Your complete BIB file should appear as shown:

```
MODULES

RegApp.exe   $(_FLATRELEASEDIR)\RegApp.exe                NK

FILES

Greenstone.bmp        $(_FLATRELEASEDIR)\Greenstone.bmp       NK
```


- Select **File** | **Save RegApp.bib** to save the changes

Notice the Syntax Highlighting and Intellisense for the BIB file that is new to CE 6.

The BIB file will cause Greenstone.bmp to be included in the image from the Flat Release Directory. You will now edit a Batch File to copy the bitmap to the Flat Release Directory whenever the subproject is built.

In the Solution Explorer:

- Double-click **postlink.bat** under the RegApp Subproject
- Add the following line (add as a line, with no returns)

```
Copy "C:\Labs\Lab1\LabFiles\Greenstone.bmp"
"%_PROJECTROOT%\cesysgen\oak\target\%_TGTCPU%\%WINCEDEBUG%"
```

- Select **File** | **Save postlink.bat** to save the changes

3.5 Build the OS Design

To build the operating system:

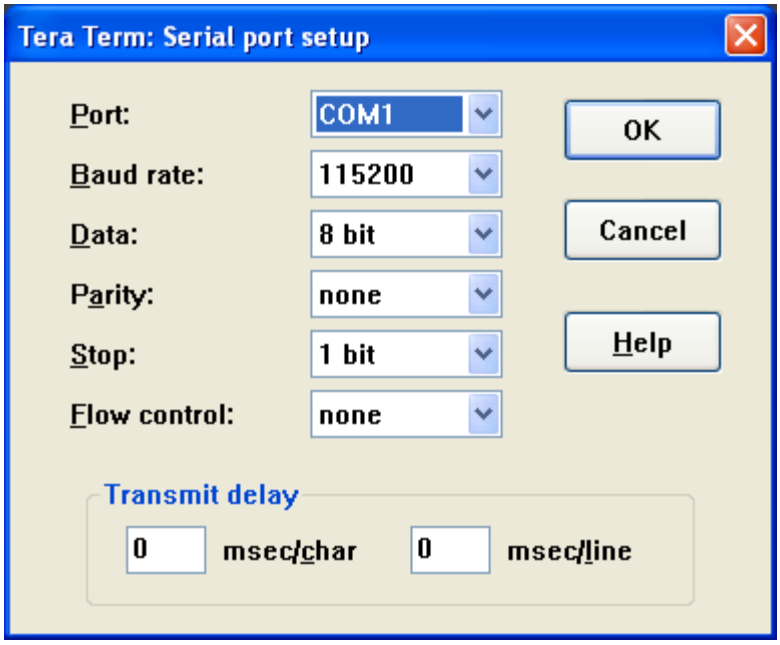
- Select **Build** | **Build Solution** – note that this could take more than 10 minutes to complete.

4 Download the Platform Image

4.1 Configure Download/Debug Transport

You are now ready to download and debug the Windows CE operating system image – before you download let's check to make sure the debug and kernel transports are configured and that we can view debug information over the serial terminal.

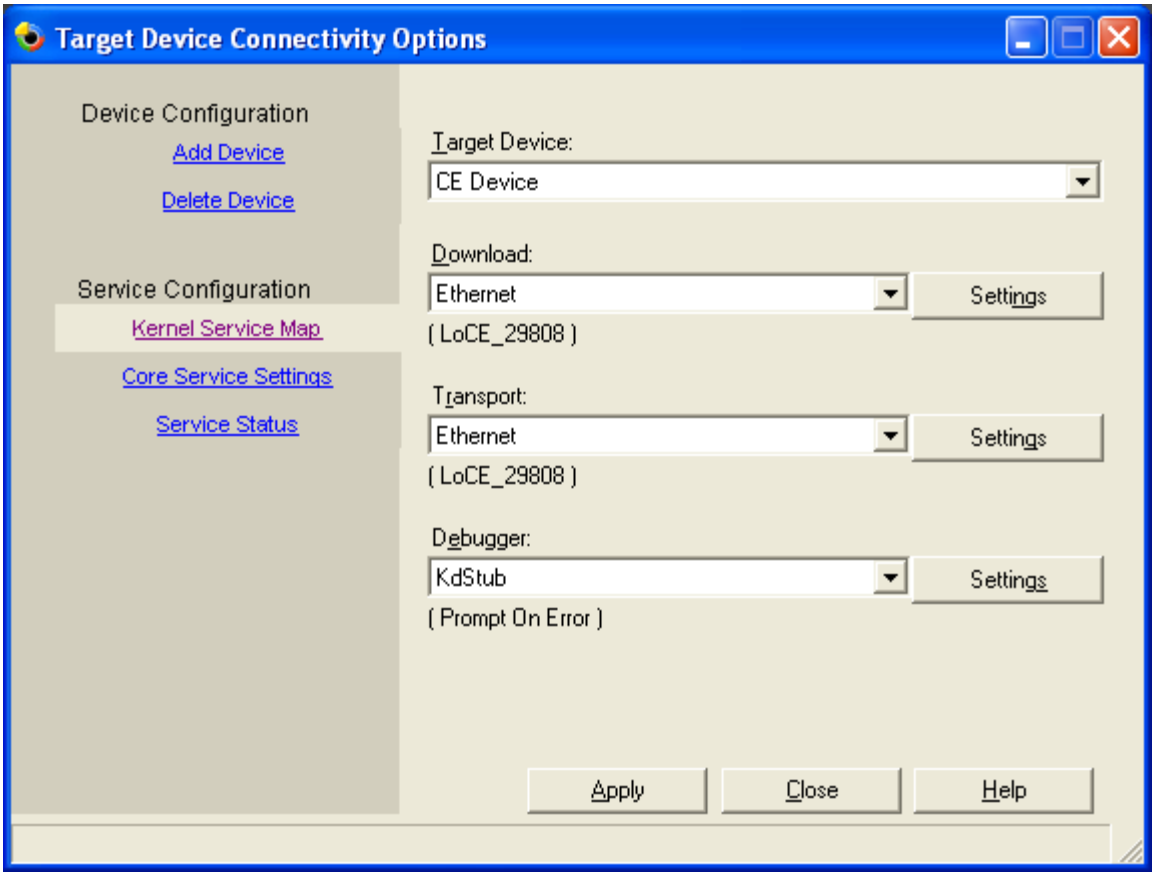
- Open **TeraTerm** by double clicking the icon on the desktop.
- Verify the serial port settings by clicking the **Setup** | **Serial Port** menu item (Your physical Port may differ):



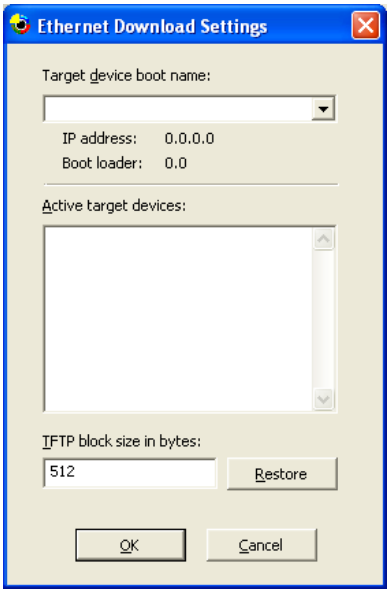
Ok now that you have configured your serial terminal we can continue to setup the device and Platform Builder. Switch back to your Visual Studio 2005 instance.

- Select **Target** | **Connectivity Options**

The following dialog will be displayed:



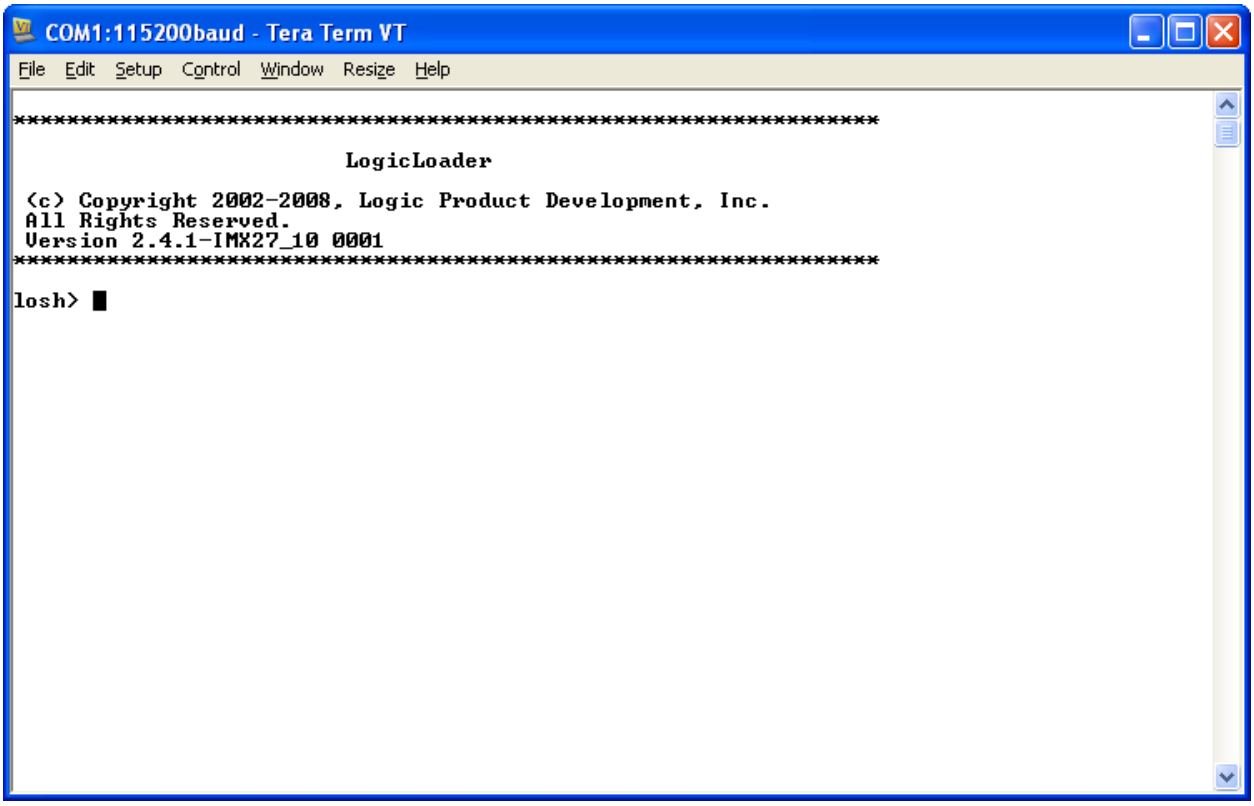
- In the **Download** box select **Ethernet**
- In the **Transport** box select **Ethernet**
- In the **Debugger** box select **KdStub**
- Click on the **Settings** box next to the **Download**: dropdown box. A box like this will appear:



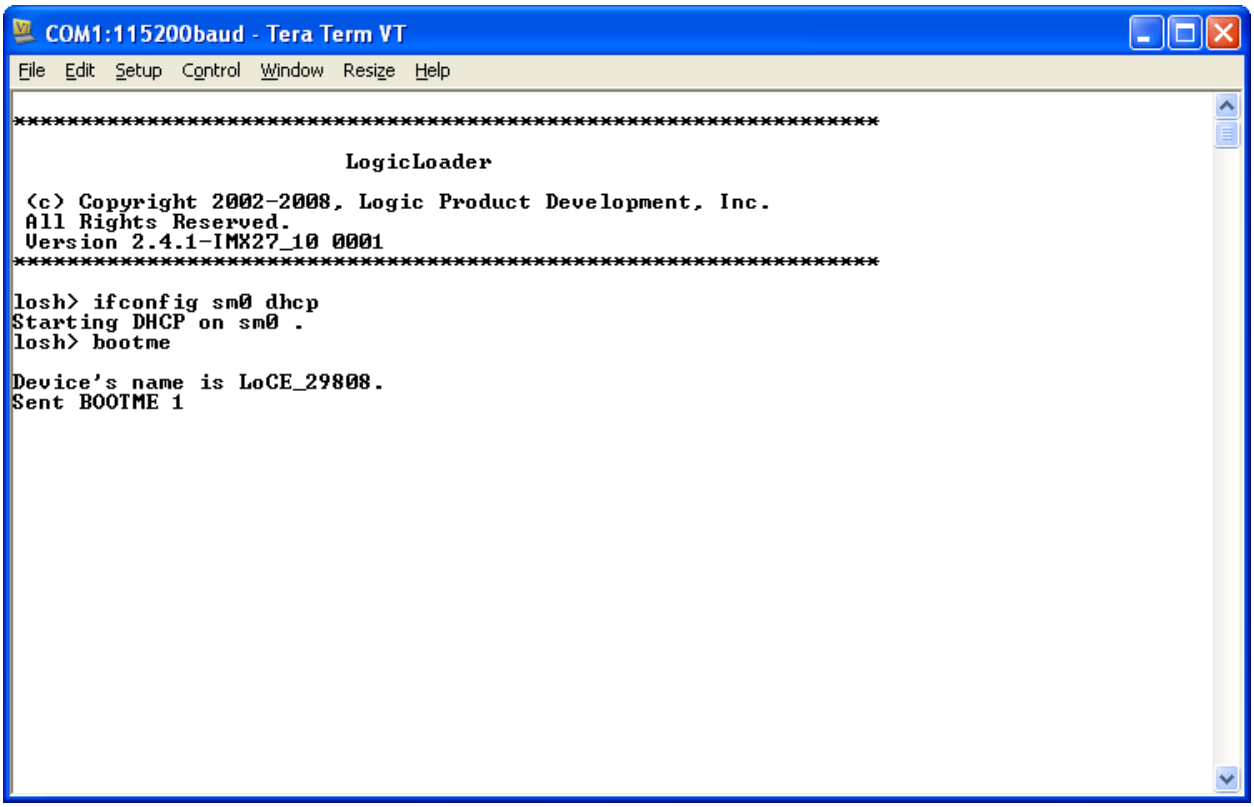
- Leave this box open and move on to the next step

Now we will switch to the **Tera Term** window to configure our device.

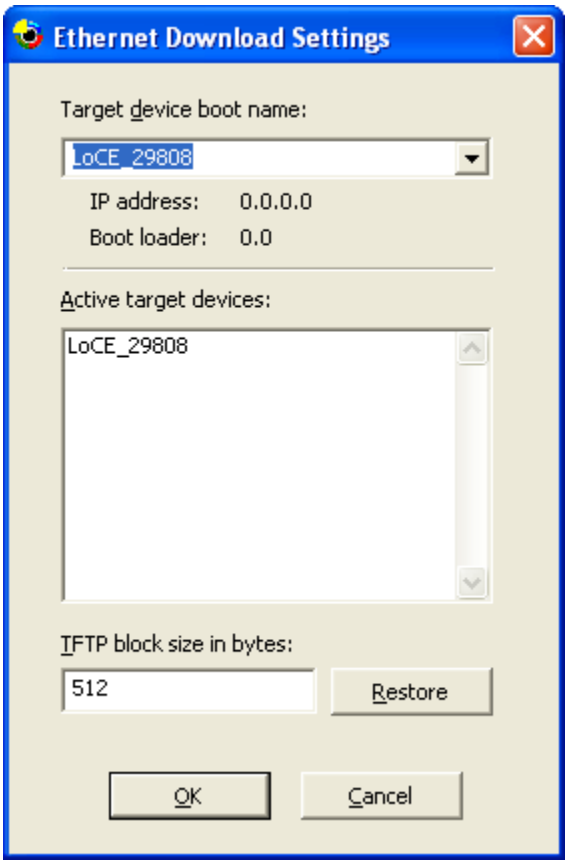
- Power on the device.
- You should see a **losh** prompt display.



- Configure the network interface using either DHCP or a statically assigned address:
  - For DHCP, type: **ifconfig sm0 dhcp**
  - For a static ip address, use the following command: **ifconfig <interface> <ip> <netmask> <gw>**
    - Example: **ifconfig sm0 192.168.1.1 255.255.255.0 192.168.1.0**
- Type the command **bootme** and press the Enter key. This will allow visual studio to connect to this device. Note the name of the device that is displayed:



- Switch back to Visual Studio and the box you left open should now be populated with a device. If there are multiple devices in the list, click on them and view the IP address. Make sure it matches the one that you wrote down.
- **Note**: If the device does not appear in Platform Builder's "Ethernet Download Settings" window, please check your firewall settings and anti-virus software. The "bootme" protocol uses raw UDP packets for communication between the device and the tools.



- Click **OK** in the small box, and then click **Apply**. Once you have done this, click the **Close** button.

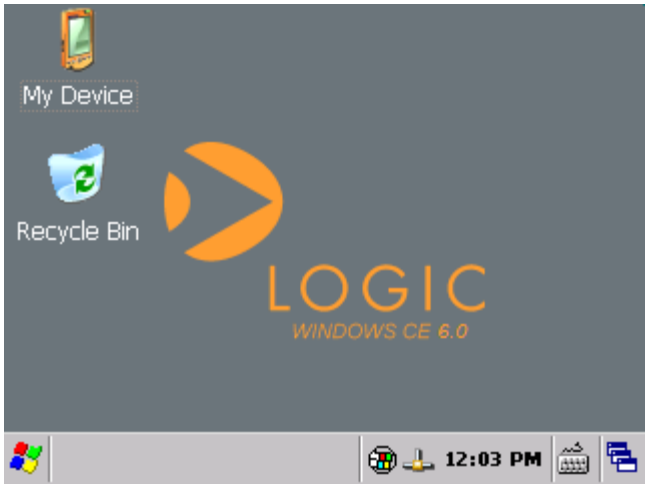
4.2 Download the Operating System

- Select **Target** | **Attach Device**
- The device should still be executing the **bootme** command, but if it is not, type **bootme** at the **losh** prompt
- When the image is finished transferring (a **losh** prompt will display), type the following command to boot the image (all on one line): **exec rtc:rtc_imx31_pmic:dbg_leds:1:disp_num:3:kitl:true:share_eth:1:**
- The debugger may pause with a **DEBUGCHK** message. In the **Debug** menu, click **Start** or press the **F5** key on the keyboard to continue execution.

The image will load quite slowly because we are in a DEBUG build.

5 Using the RegApp Application

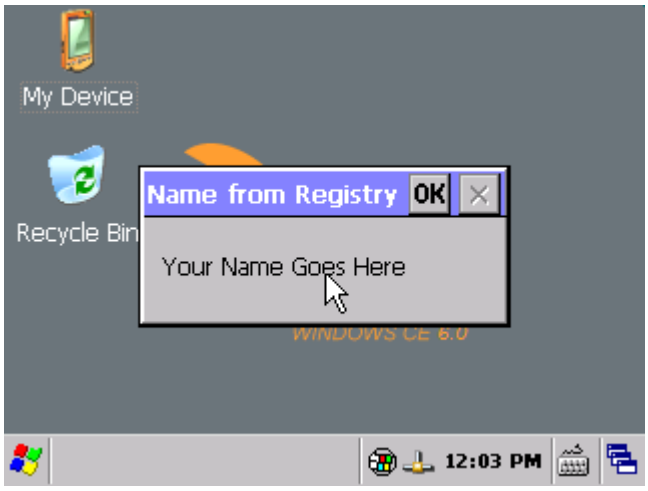
Here is an example of how the Windows CE Operating System you built looks:



You can use Target Control to remotely launch the RegApp application on the device.

In Visual Studio:

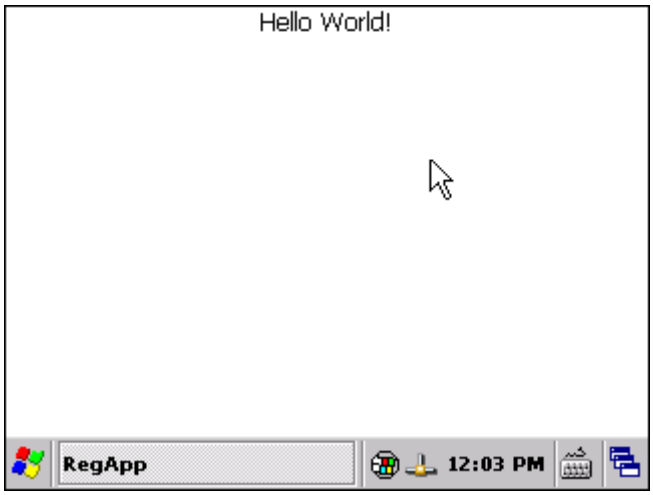
- Open **Target** | **Target Control**
- At the target control command prompt, type: `s RegApp` (this is the command used to start/launch the RegApp application – you could also use the **Target** | **Run** menu option)
- Look on your device to see that the application has run.



The contents of the registry key you added to the platform will be displayed in a message box.

- Click **OK** to close the MessageBox

You have successfully modified the registry before building the Operating System. In this case you were looking for a specific key from the registry which is not part of the default operating system image. With Windows CE 6, user created projects wrap up all the information needed to build in a standalone 'atomic' Subproject. Each Subproject can define its own folders, registry information, files, databases, and platform dependencies.



You can also disconnect Platform Builder from the target.

Within Visual Studio:

- Select **Target** | **Detach Device**
- Click **Yes** to stop the debugger

6 Image Explorer

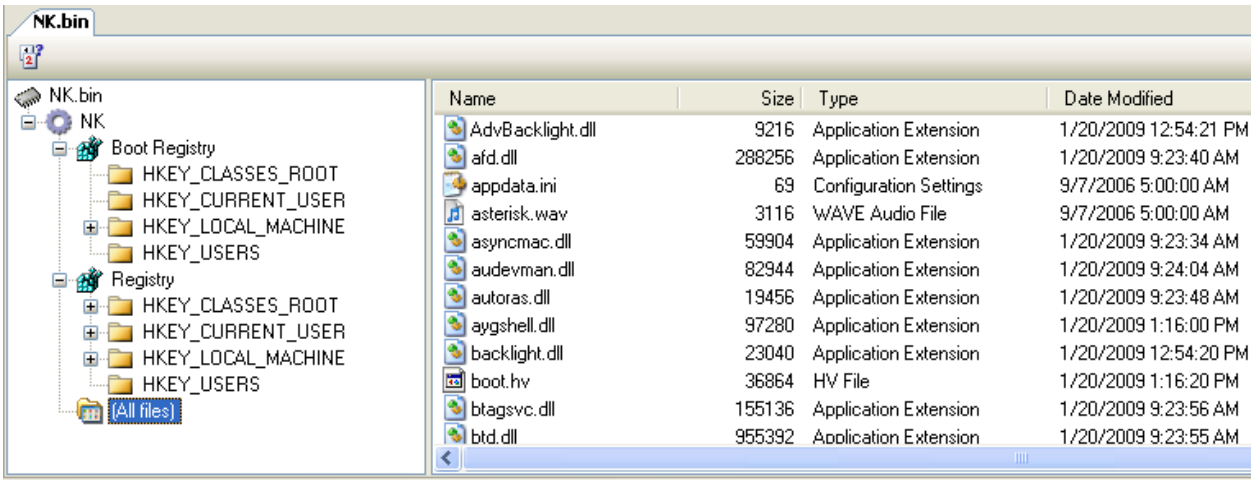
Windows CE 6 introduces the new “Image Explorer” feature. The Image Explorer allows you to look inside an nk.bin Windows CE OS image and analyze its contents. You will use Image Explorer to look inside the nk.bin that you just built and debugged.

6.1 Open the Image

Within Visual Studio:

- Select **File** | **Open** | **File…**
- Change the **Files of type** filter to **Windows CE Run-Time Images**
- Open the file **C:\WINCE600\OSDesigns\OSDesign1\OSDesign1\RelDir\TrainingBSP_ARMV4I_Debug\NK.bin** (You may have to change the **Files of Type**: to **All Files (*.*)** to see the **NK.bin** file

The nk.bin file is opened as an Image Explorer document. Notice that you can browse the contents of both the Registry and the File System.



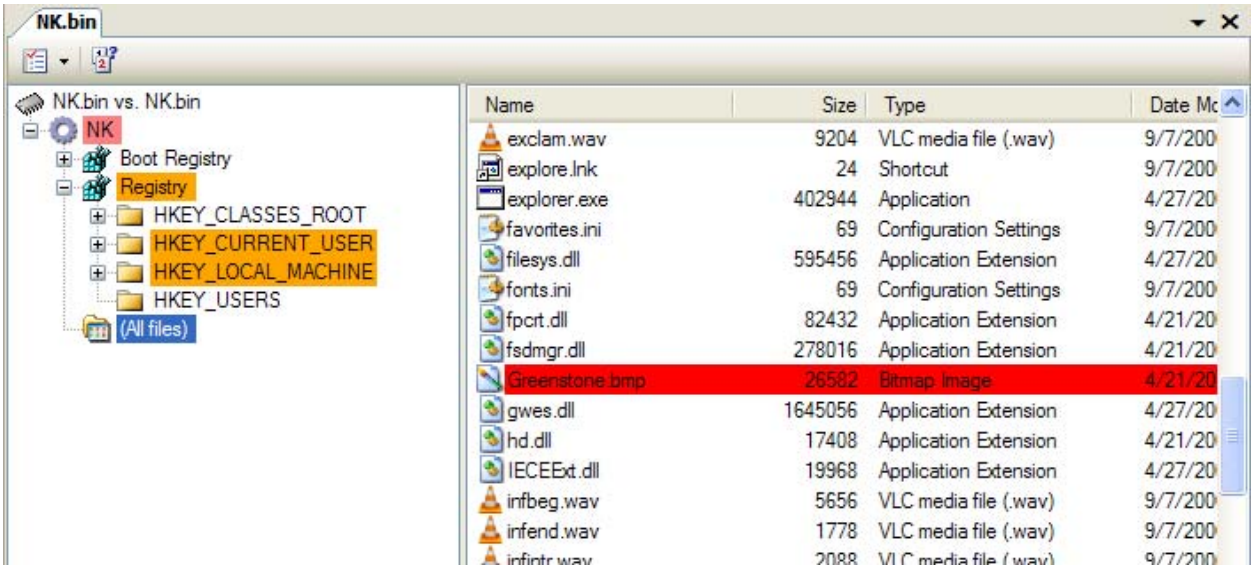
The Image Explorer also allows you to compare two OS Images. You will now open a baseline Mobile Handheld Image and compare it with the one that you just built.

- Click the **Compare to Other Image** ( ) button.
- Open the file **C:\Labs\Lab1\LabFiles\NK_base.bin**

The Image Explorer depicts the relationship between the two image files. Items that are only in the original image are shown in Red. Items that are only in the second image are shown in Yellow. Items in both images are shown in White.

If you browse through the tree, you will discover the differences between the images:

1. Greenstone.bmp which you included via a BIB file
2. The registry key you set via a REG file
3. RegApp.exe
4. The included drivers from the BSP



- Select the **[All Files]** node
- Browse for **Greenstone.bmp**

- Double-click **Greenstone.bmp**

The bitmap is extracted from the nk.bin file and opened for viewing.

7 Edit the Catalog

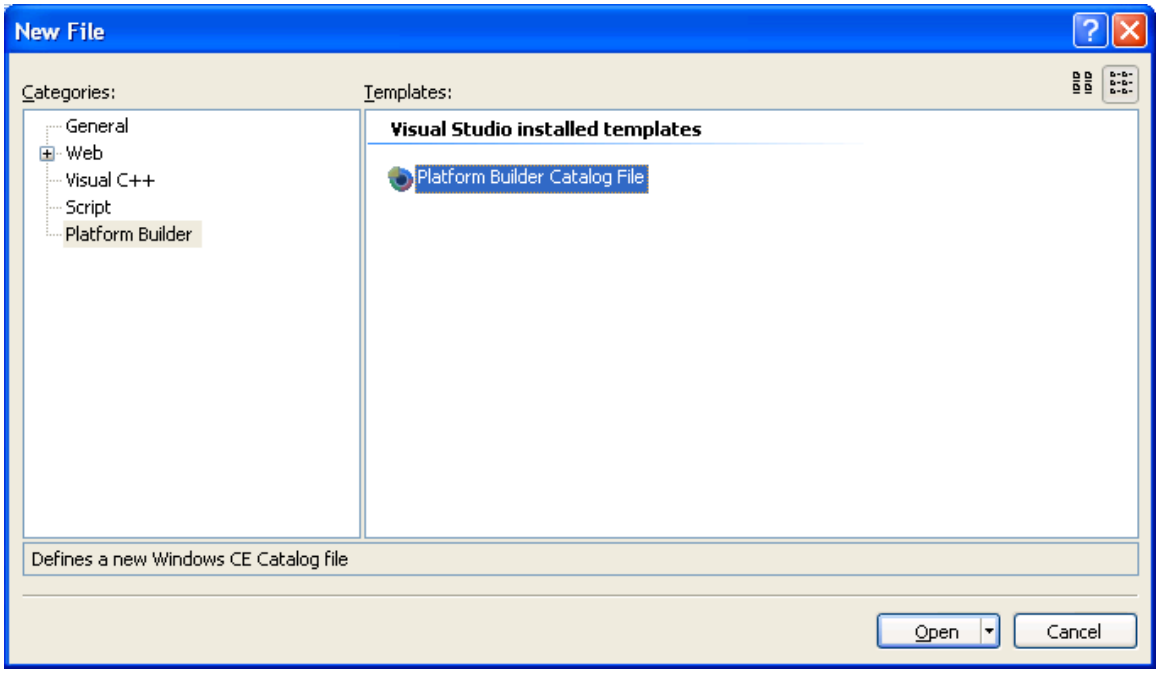
Windows CE 6 no longer uses CEC files to define the Catalog. The Catalog is now made up of a series of XML files (PBCXML). These files are loaded and parsed on the fly, so no extra “import” step is needed as in the past.

You will use the Catalog Editor to create a new PBCXML file that adds your RegApp subproject to the Catalog.

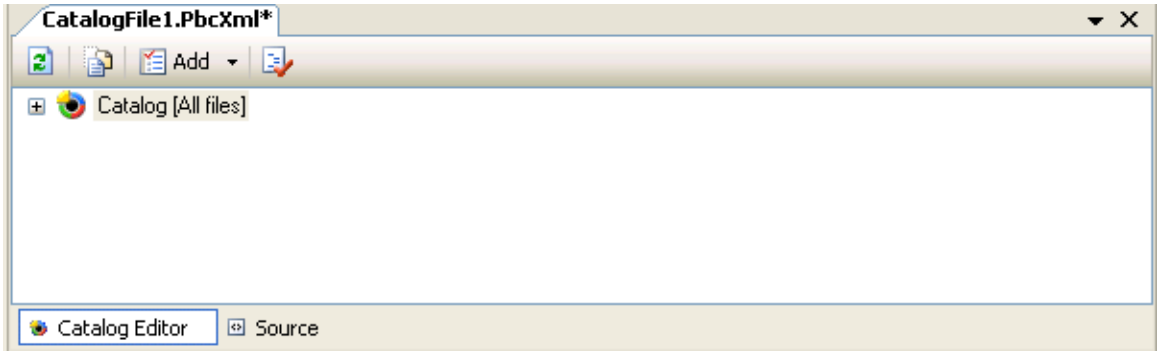
7.1 Create a new Catalog File

In Visual Studio 2005:

- Select **File** | **New** | **File…**
- Select the **Platform Builder** category

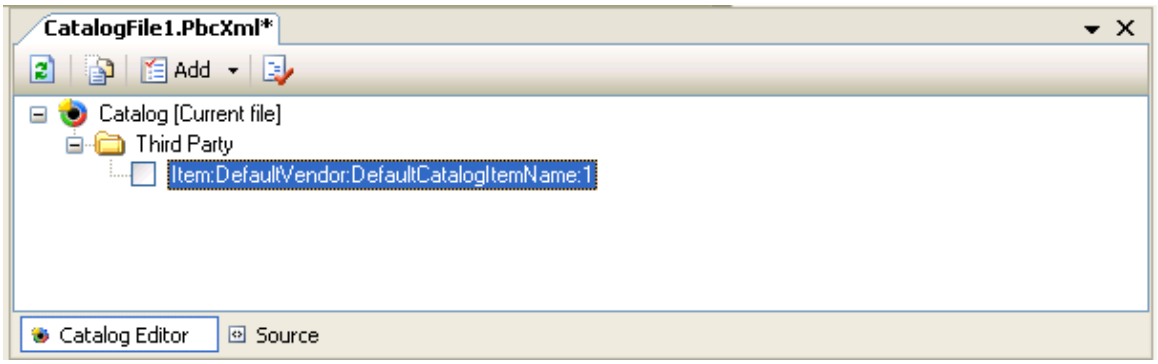


- Choose to create a new **Platform Builder Catalog File**



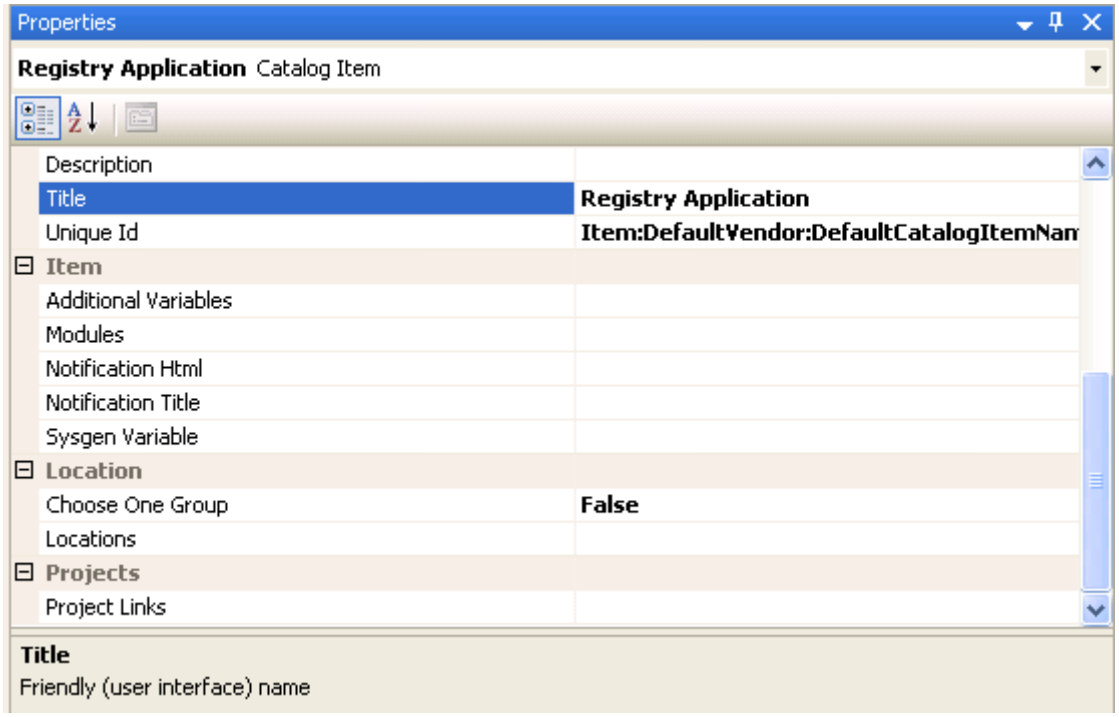
- **Right-click** the Catalog node

- Select **Add Catalog Item**

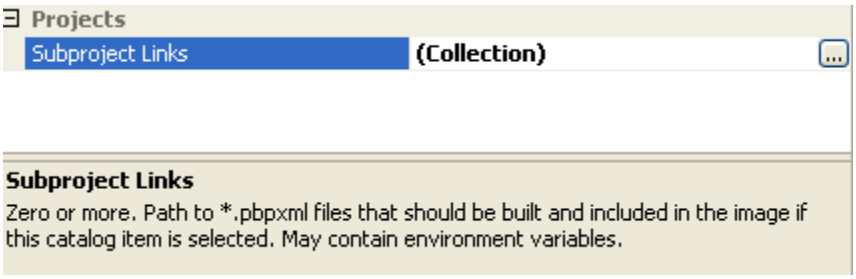


From here, you can change the name of your new Catalog Item via the Properties Pane

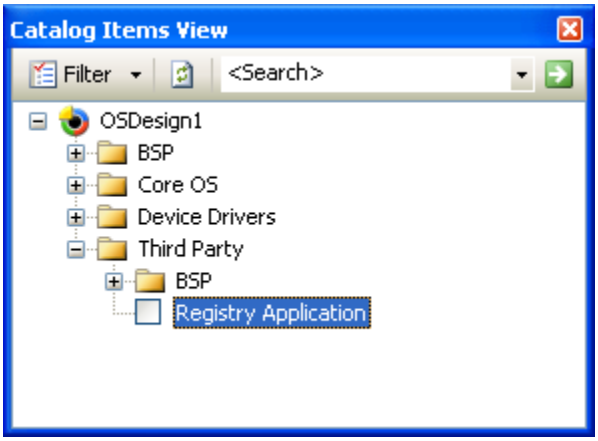
- **Right-click** the new Catalog Item
- Select **Properties**



- Change the value of the Title field in the Properties Pane to **Registry Application**
- Click on the **Subproject Links** field of the Properties Pane
- Click the "…" button to bring up the editor



- Choose **Add**
- Open the file: **C:\WINCE600\OSDesigns\OSDesign1\OSDesign1\RegApp\RegApp.pbpxml**
- Select **OK**
- Click on the **Registry Application** node in the Catalog Editor
- Select **File** | **Save CatalogFile1.pbcxml**
- Save the file to **C:\WINCE600\PUBLIC\Common\Catalog\CatalogFile1.pbcxml**
- Select **View** | **Other Windows** | **Catalog Items View**
- Click the **Refresh** button in the catalog view (The button next to the Search box)



Notice that your catalog item has been added to the Catalog! You can now easily add the same subproject to another OS Design by selecting this catalog item.

8 Implementing a Driver

In this exercise you will implement a driver as a subproject.

The first thing we will do is add a Driver subproject to our OSDesign.

- Copy the **LEDDriver** folder from **C:\Labs\Lab1\LabFiles** into your **OSDesign** folder at **C:\WINCE600\OSDesigns\OSDesign1\OSDesign1**.
- In the Solution Explorer window right click on the **Subprojects** folder and select **Add Existing Subproject…**
- Navigate to the **LEDDriver** folder that you just copied.
- Select **LEDDriver.pbpxml** and click **Open**. Visual Studio will add the project to your current solution.
- Open the **LEDDriver** folder in Windows Explorer.

- Open **LEDDriver.def** in Visual Studio. This file contains the entry points that other functions/applications will use to access the driver.
- Open **LEDDriver.reg** in Visual Studio. Expand the nodes and examine the values. The .reg file contains information in keys about the driver, how it is named, what indexes it can take, etc.

The registry also tells us that the driver is not automatically loaded, because the registry key is not in the built-in folder.

- Open **LEDDriver.bib**. As you saw before, .bib files are important to the makeimage step in the build process. They are used to track the memory layout.
- Open the **sources** file in Visual Studio. If you are opening files from Visual Studio, double clicking on the **LEDDriver** subproject will open the sources file. The sources file determines whether the file will be a dll, lib, exe, etc. as well as bringing in any includes that the file needs, and specifying where the file will be located.
- Close the files. These files make up the dll of the driver.
- Locate and expand the **LEDDriver** subproject in the **Subprojects** node of the Solution Explorer.

Now we will configure the subproject image settings. We will configure the subproject settings so that we can easily debug it without needing to rebuild our OS design.

- Right click on the **OSDesign1** project in the Solution Explorer and select **Properties**.
- From the **Configuration** drop down select **All Configurations**.
- Expand the **Configuration Properties** node and select **Subproject Image Settings**.
- Double-click the **LEDDriver** entry in the **Project settings in run-time image** box. The **Edit Run-Time Image Settings** dialog box will appear.
- Select the **Always build and link as debug** check box, and click **OK**. Make sure that the **Exclude from image** box is **NOT** checked.
- Click **OK** on the **OSDesign1** Property Pages dialog.
- Expand the **Source** files node of the **LEDDriver** subproject.
- Double-click the **LEDDriver.cpp** file. The file will load in the Visual Studio editor.
- Navigate to the **BOOL LED_IOControl** function near the bottom of the file. Insert the following lines where shown.

```
DEBUGMSG(1, (TEXT("LED on\r\n")));

DEBUGMSG(1, (TEXT("LED off\r\n")));
```

LEDDriver.cpp

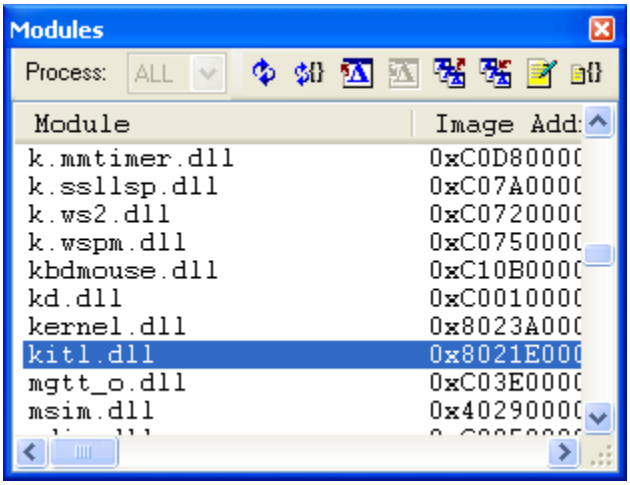
(Global Scope) | LED_IOControl(DWORD hOpenContext, DWORD dwCode, PBY

```
// Perform the action ....
switch (dwCode)
{
    case IOCTL_LED_ON:
    {
        // Check parameters
        if (pBufIn != NULL && pBufOut != NULL)
        {
            // Turn On the LED
            DEBUGMSG(1, (TEXT("LED on\r\n")));
            bResult = TurnLEDOnOff(pLEDInit,TRUE);
        }
        else
        {
            bResult = FALSE;
        }
    }
    break;

    case IOCTL_LED_OFF:
    {
        // Check parameters
        if (pBufIn != NULL && pBufOut != NULL)
        {
            // Turn Off the LED
            DEBUGMSG(1, (TEXT("LED off\r\n")));
            bResult = TurnLEDOnOff(pLEDInit,FALSE);
        }
        else
        {
            bResult = FALSE;
        }
    }
    break;
```

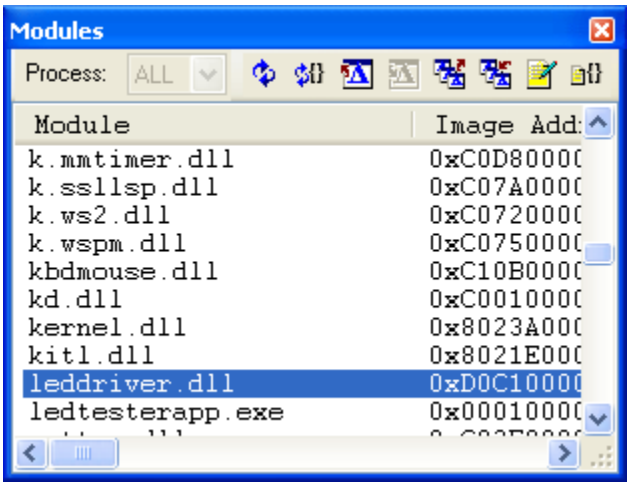
- Save **LEDDriver.cpp**. The **DEBUG** messages will now display whenever the LED is turned on or off.
- Right click the **LEDDriver** subproject and select **Build**. The build should succeed with 0 errors. A **Makeimg** will also be performed.
- Copy the **LEDTesterApp** folder from **C:\Labs\Lab1\LabFiles** into your **OSDesign** folder at **C:\WINCE600\OSDesigns\OSDesign1\OSDesign1**.
- In the Solution Explorer window right click on the **Subprojects** folder and select **Add Existing Subproject…**
- Navigate to the **LEDTesterApp**folder that you just copied.
- Select **LEDTesterApp.pbpxml** and click **Open**. Visual Studio will add the project to your current solution.
- Right click the **LEDTesterApp** subproject and select **Build**. The build should succeed with 0 errors. A **Makeimg** will also be performed.

- Now we will attach our device and see that the driver is not loaded upon startup. The driver will be loaded, however, when we run the LEDTesterApp. This program will use an API to load the driver.
- Go to **Target** | **Attach Device**.
- Reset your device and load the image.
- When the image has loaded on the device go to **Debug** | **Windows** | **Modules** and see if you can locate the LEDDriver.dll in the list. It should not be there:



- Now go to **Target** | **Run Programs** | **LEDTesterApp.exe**

The LEDTesterAPP.exe loads the LEDDriver.dll. You can now see the DLL listed in the Modules list!



You can view the debug output from this application in the Windows CE debug output window.

- Select **View** | **Output**
- Make sure that the Show output from selection is **Windows CE Debug**
- Click in the output window and press **ctrl-f**
- Search for **LED on**
- You should see instances of this as well as **LED off**
- Select **Target** | **Detach Device**

9 Integrate a Driver into the BSP

In this exercise, we will integrate the driver into the BSP so that it loads automatically at startup.

First we need to remove the LEDDriver from our BSP. We are removing it because we will be adding it as a driver that is part of the BSP rather than just a subproject.

- Right click on the **LEDDriver** subproject and select **Remove**.

Now we will add the source code of the driver to the BSP

- Move the **LEDDriver** directory from **OSDesign** folder to the **C:\WINCE600\PLATFORM\TrainingBSP\SRC\DRIVERS** directory.
- Delete the **obj** folder under **LEDDriver** if it exists.
- Double-click the **C:\WINCE600\PLATFORM\TrainingBSP\src\drivers** node in the Solution Explorer. This will open the **Dirs** file.
- Add the following line to the end of the **Dirs:**

```
LEDDriver\
```

- Save and close the **Dirs** file.

Next we need to edit the Drivers SOURCES file. This is a file that is read by the compiler at build time. It can be used to specify driver entry points, link to other libraries and control the build of the driver through set environmental variables.

- Expand the **DRIVERS** node.
- Double-click on the **LEDDriver** node. This should open the sources file for editing.
- Find the `RELEASETYPE=LOCAL` line. Change **LOCAL** to **PLATFORM** so the line looks like this: `RELEASETYPE=PLATFORM`
- Save and close the **sources** file.

The **RELEASETYPE** parameter is used by the build engine to identify where to put the files resulting from the build. **PLATFORM** means **BSP** folder, the driver **dll** will be stored in **target**\ folder.

Now we will update the driver's registry file. Here we can specify where the driver will load and also provide any specific parameters that the driver may use.

- Expand the **Parameter files** node under **LEDDriver** in the Solution Explorer window. Open the **LEDDriver.reg** file.
- Click on the **Source** view at the bottom of the window. Change the following line: `[HKEY_LOCAL_MACHINE\Drivers\LEDDriver]`
  to: `[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\LEDDriver]`
- **Save** and close the **LEDDriver.reg** file.

Note: All drivers declared under **BuiltIn** will be loaded at startup.

Next we will add our driver as a component in the BSP's catalog. This way when you select the catalog item the driver will be built and included in the BSP's run-time image.

- From the Visual Studio menu, select **File** | **Open** | **File…** and navigate to the **C:\WINCE600\PLATFORM\TrainingBSP\CATALOG** folder.
- Change the file mask to show Files of type: **All Files (*.*)**
- Open the **TrainingBSP.pbcxml** file.

- Expand the catalog **Third Party** tree to until you see the **Device Drivers** node.
- Right click on the **Device Drivers** node and select **Add Catalog Item**.
- In the **Properties** window, set the **Description** field to **LED Driver**.
- Set the **Title** field to **LED Driver**.
- Set the **Unique Id** field to **Item:GeneriCo:LEDDriver**.
- Click in the **Additional Variables** field. Then click the box with the "…" at the right end of the field. This will take you to the **String Collection Editor** dialog box. Type **BSP_LEDDriver** and click **OK**.
- In the same way, set the **Modules** field to **LEDDriver.dll**.
- **Save** and close the file.

Since we made the driver available in the catalog we will need to add the information to the BSP so that the driver files will be included in the build.

- Open the **platform.bib** file in the **Parameter Files** node of the **TrainingBSP** in the Solution Explorer under the **PLATFORM** node.
- Add the following lines near the top of **platform.bib** as the first entry in the MODULES section:

```
IF BSP_LEDDriver

   LEDDriver.dll              $(_FLATRELEASEDIR)\LEDDriver.dll
NK SHK

ENDIF BSP_LEDDriver
```

When you are done, the top of the file should look similar to the following:

```
MODULES



;  Name                 Path
Memory Type

;  --------------       ----------------------------------      --------
---



IF BSP_LEDDriver

   LEDDriver.dll              $(_FLATRELEASEDIR)\LEDDriver.dll
NK SHK

ENDIF BSP_LEDDriver



;-------------------------------------------------------------------------
--------

; GWES drivers

;

; @CESYSGEN IF CE_MODULES_GWES
```



```
;
; Display driver
;
; @CESYSGEN IF CE_MODULES_DISPLAY
```

- **Save** and close the file.
- Now we will do the same as above, except it will be for including the driver's registry file information into the run-time image.
- Open the file **Platform.reg** from the **Parameter Files** node of the **TrainingBSP** using the Solution Explorer.

Note: The non-key entries in Windows Registry are case-sensitive.

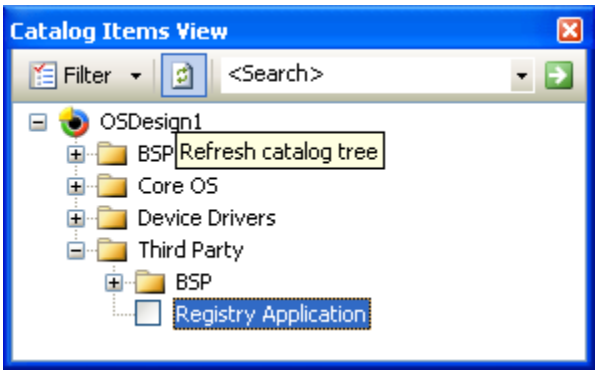
- Click on the **Source** button and the bottom of the window.
- At the bottom of the file add the following code

```
; ------------------------------------------------------------------------
; LED Driver
;
IF BSP_LEDDriver
#include "$(_TARGETPLATROOT)\SRC\DRIVERS\LEDDriver\LEDDriver.reg"
ENDIF BSP_LEDDriver
; ------------------------------------------------------------------------
```

- **Save** and close the file.

Now we can select the driver in the catalog to include it in our run-time image

- Switch to the **Catalog Items View**.
- Refresh the **Catalog Items View** by clicking on the refresh button located on the command bar.



- Expand the **TrainingBSP: ARMV4I** node under **BSP** (**BSP** is under **Third Party**).

Note: Ensure that the **Filter** option is set to **All Items in the Catalog**. The **Filter** option is a drop down box in the upper right hand corner of the **Catalog Items View**.

- Select the **LED Driver** item under **Device Drivers**.

- Select **Build** | **Advanced Build Commands** | **Build Current BSP and Subprojects** from the Visual Studio menu. This may take a few moments.

We should verify that the driver made it into the image. We can do this using the Image Viewer tool we previously learned about.

- From the Visual Studio menu, select **File** | **Open** | **File…** and navigate to the **C:\WINCE600\OSDesigns\OSDesign1\OSDesign1\RelDir\TrainingBSP_ARMV4I_Debug** folder.
- Open the **NK.bin** file. This will bring up the **Run-Time Image Viewer**.
- Click on the **(All Files)** node in the **Image Explorer**. This shows all files that are built into the OS run time image.
- Verify that **LEDDriver.dll** is listed.
- Verify that the **[HKLM\Drivers\BuiltIn\LEDDriver]** key exists under the registry node.
- Close the **Image Viewer**.

We need to modify our LEDTesterApp so that it does not load the device driver.

- In the Solution Explorer window open **LEDTesterApp.cpp**.
- Near the top of the file comment out the following line: `#define LOAD_DRIVER` it should turn green and look like this: `//#define LOAD_DRIVER`
- **Save** and close the file.
- Right click on the subproject and select **Build**.

Lastly we will test the driver

- Click **Target** | **Attach Device**
- Reset your device and load the image.
- Once the image has booted go to **Target** | **Run Programs** | **LEDTesterApp.exe** and make sure it runs correctly.

Congratulations! You have successfully integrated your driver into the BSP!

10 Summary

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

- Created an OS Design
- Customized the OS Design by adding applications
- Added custom Registry information to the Operating System
- Included an external file into the operating system image
- Built an OS image
- Downloaded an operating system to hardware
- Used Target Control to launch an application on the device
- Examined the target registry with our own custom application
- Used the Image Explorer to look inside and compare two nk.bin files
- Added an item to the catalog using the Catalog Editor
- Created a simple driver

- Added that driver to the BSP
- Tested the driver with a sample test application

