Applieddata.net

Embedded Computer Systems

Developer's Guide
C# Edition

ADS document # 110010-10071-3

PRELIMINARY

Applied Data Systems

9140A Guilford Road
Columbia MD 21046
301-490-4007

www.applieddata.net

© 2003 Applied Data Systems

(This page intentionally blank)

PRELIMINARY

Revision History

LTR	DESCRIPTION	DATE	BY
-	Create Date	Aug 4, 2003	ctacke
1-2	Prelim 2 release	Aug 14, 03	ctacke
1-3	Added Install info Added section numbering Added BitsyX Memory Map	Sep 18, 03	ctacke

PRELIMINARY

(This page intentionally blank)

PRELIMINARY

Table of Contents

PRELIMINARY

PRELIMINARY

1 Getting Started

This manual provides information about developing applications fpr the Windows CE.NET (or CE 4.x) operating system on Applied Data Systems (ADS) devices.

1.1. Document Organization

This document is organized into seven sections. The sections are designed to logically guide you through the process of creating and debugging an application on your Applied Data Systems development system.

Section 1 provides a brief outline of the development tools you will need as well as other documentation that you will find useful in your development effort.

Sections 2 and 3 cover the process of creating and debugging an application for your device.

Section 4 briefly describes some of the desktop tools available to aid in your development effort.

Section 5 provides a look at many ways in which you can adapt your ADS development system to specifically meet your production needs once you have your application ready.

Sections 6 and 7 are reference sections covering the ADS system drivers and the Windows CE boot process as it specifically relates to ADS devices.

1.2. Windows CE Development Tools

You will need to install the following tools on your development PC **and** in the order described.

Microsoft ActiveSync

Microsoft ActiveSync is a communications conduit that allows a Windows CE device and connected PC. As of this writing, the latest version is Version 3.7 and it can be downloaded directly from Microsoft's web site at:

http://www.microsoft.com/windowsmobile/resources/downloads/pocketpc/activesync37.mspx

Microsoft Visual Studio 2003

The Development Environment for C# is Microsoft Visual Studio 2003. You must have a full version of Studio, the single-language versions do not have the tools necessary for Smart Device Programming, which is necessary for CE device development. Visual Studio can be purchased from most software resellers and more information is available from Microsoft's web site at:

http://msdn.microsoft.com/vstudio/

Visual Studio 2003 Add-on Pack

Out of the box, Microsoft Visual Studio 2003 only has connectivity tools for Pocket PC devices. To be able to connect to CE 4.x devices, including all Applied Data Systems devices, you must install the Visual Studio 2003 Add-on Pack. The Add-on Pack is freely downloadable from Microsoft's web site at:

https://www.microsoft.com/windows/embedded/ce.net/downloads/

PRELIMINARY

1.3. Additional Reading

ADS Developer's Getting Started Guide for CE

http://www.applieddata.net/forums/topic.asp?TOPIC_ID=609
ADS Document 411010-1104x

If you have not already done so, we highly recommend that you download and read the Developer's Getting Started Guide. This guide covers introductory topics such as your development system inventory, making an ActiveSync connection between the device and your PC and making use of device's debug port.

Microsoft .NET Compact Framework (Core Reference)

by Andy Wigley, Stephen Wheelwright, Mark Sutton
Microsoft Press, 2003, ISBN 0735617252

This comprehensive reference provides developers with the information they need to develop new applications or move existing applications to handheld devices and other resource-constrained hardware. It offers specific techniques for writing mobile applications, including developing GUI elements using Web Forms, transferring data using XML Web services, working with local and remote data sources, and developing applications that can operate in a disconnected state from the wireless network. The book illustrates each technique with working code samples in Visual Basic .NET and Visual C# .NET. It also includes a quick reference appendix showing the differences between the .NET Compact Framework and the full .NET Framework

2 Creating Smart Device Assemblies

Starting with Windows CE 4.1, Microsoft introduced the Microsoft .NET Compact Framework (or CF), which is a subset of the Microsoft .NET Framework available on desktop computers. The CF provides a set of runtime components for just-in-time compiling (JITting) Microsoft Intermediate Language (MSIL) assemblies. A discussion of the framework, JITting, and MSIL is beyond the scope of this document, but there are many resources available both online and in your local bookstore.

With respect to writing Smart Device assemblies for your ADS system, there are some pieces of information that are important:

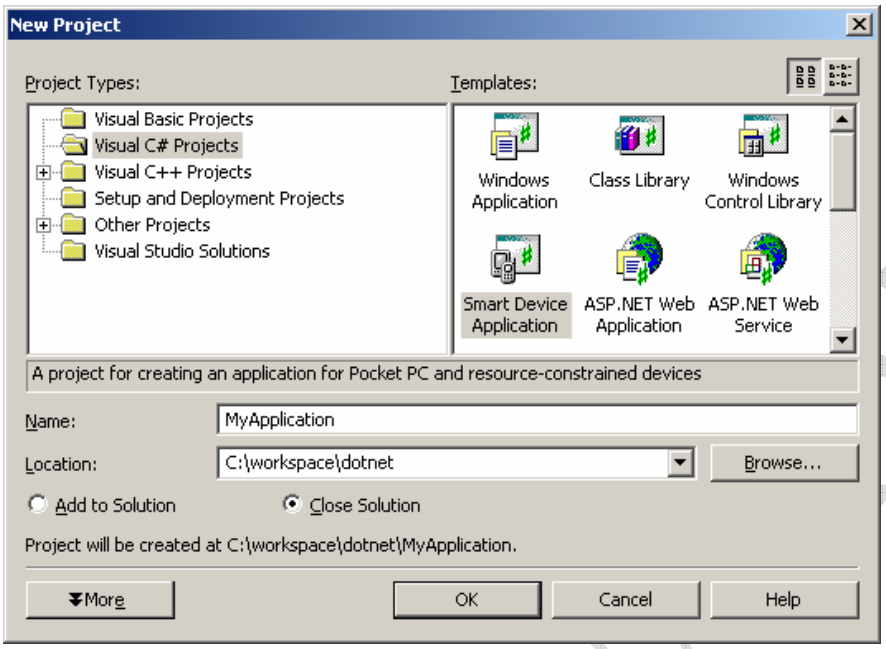
- Smart Device Assemblies are considered "managed" code. The current version of the Compact Framework supports development in only two languages: C# or VB.NET. The CF does not support managed C++.
- Smart Device Programmability (SDP) is only available using Microsoft Visual Studio 2003, Professional or Enterprise Architect. The single-language development editions do not have SDP included in them.
- Desktop assemblies built against the full Framework can not be run against the CF. However assemblies built against the CF *can* be run against the full Framework.

2.1 Creating a Smart Device Solution

To create any Smart Device assembly, whether an application or class library, you first must create a Solution and Project in Visual Studio 2003. This is done easily by using Studio's New Project Wizard.

{Add Text here}

PRELIMINARY

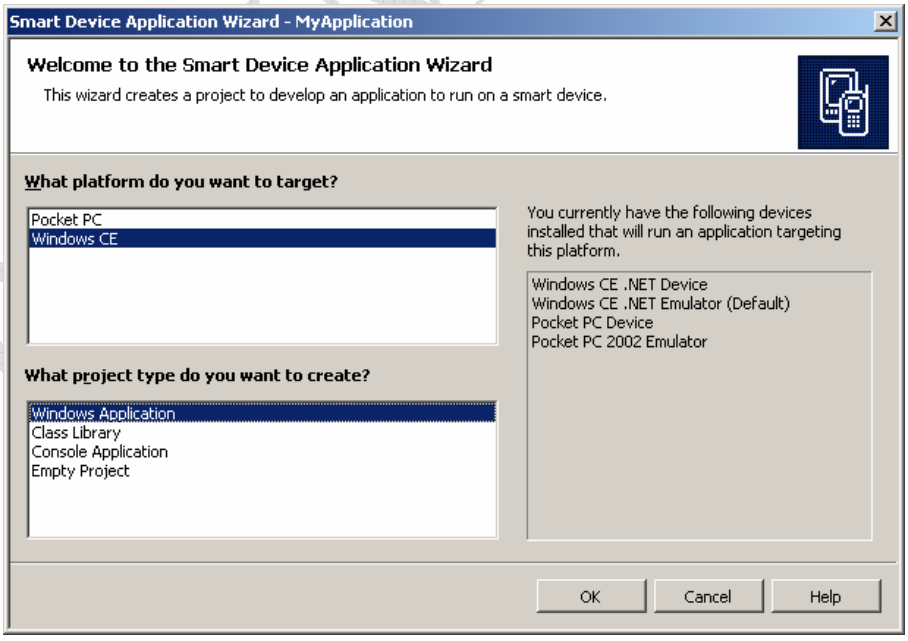


{Add Text here}

2.2 WinForms Applications

WinForms applications are the standard GUI-style application and are compiled to an executable (EXE) format.

{Add Text here}

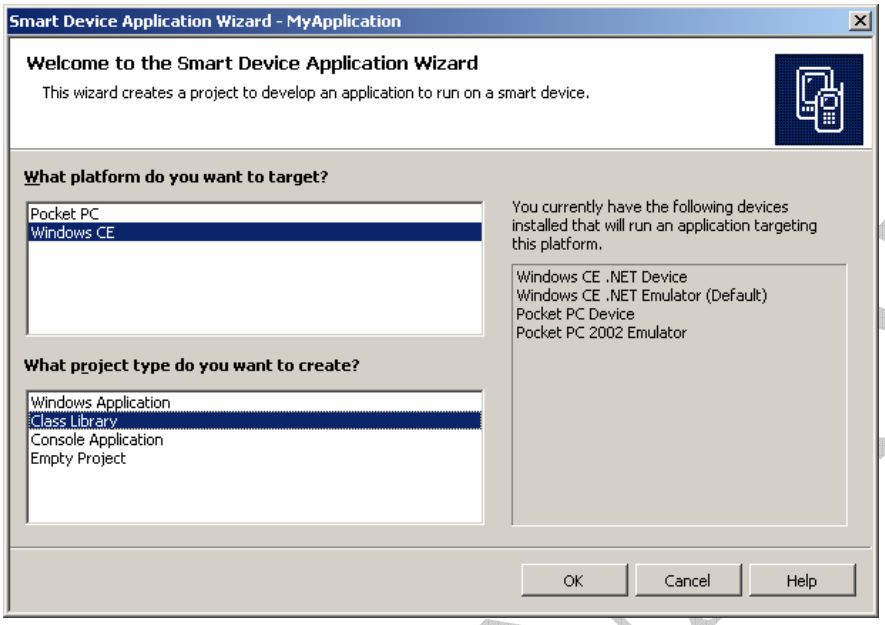


2.3 Class Libraries

Class libraries are compiled to dynamically linked library (DLL) format.

PRELIMINARY

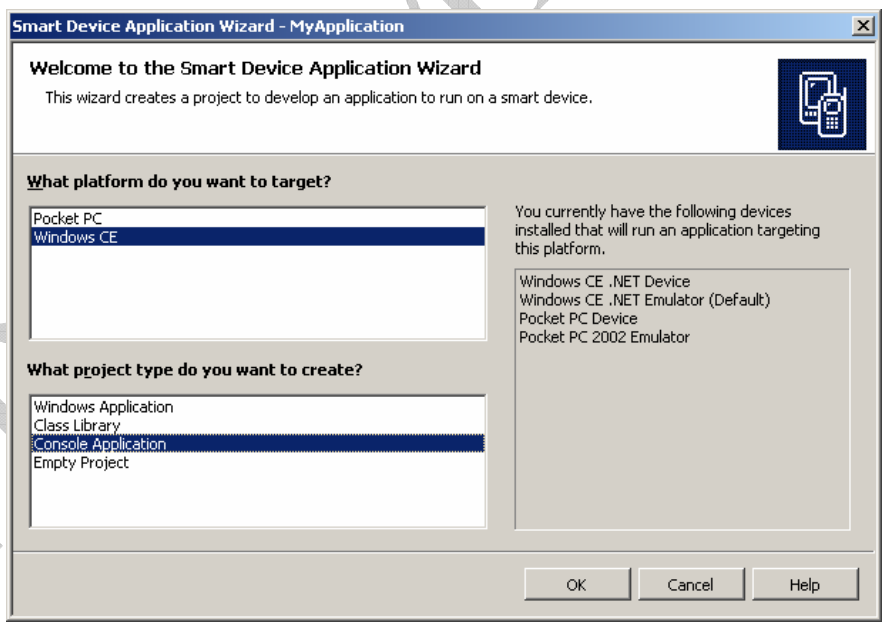
{Add Text here}



2.4 Console Applications

Console applications are compiled to executable (EXE) format.

{Add Text here}



3 Debugging Smart Device Assemblies

Debugging on CE 4.x devices requires the Visual Studio 2003 Add-on Pack (see Section 1).

PRELIMINARY

3.1 ActiveSync

Before you can debug an application on the device, you must first have an active ActiveSync connection.

For more information see the ADS Developer's Getting Started Guide for CE (see Section 1).

3.2 Connecting Studio 2003 to the Device

One of the most challenging parts of debugging a Smart Device application with Visual Studio 2003 is just making a solid connection between Studio and the device. While Pocket PCs seem to enjoy a good connectivity environment, CE 4.x devices are not so fortunate. In this section we will cover a couple methods for making the connection, as well as some troubleshooting guidelines.

Before proceeding, make sure that you have the following installed on your PC and that they were installed *in this order*:

1. Microsoft ActiveSync
2. Visual Studio 2003
3. Visual Studio 2003 Add-on Pack

If you have installed in any other order (such as upgrading ActiveSync versions after installing Studio 2003) you will not be able to connect. You will have to uninstall and reinstall Studio.

3.2.1 Step 1 – Make an ActiveSync Connection

The first step for debugging-no matter which Studio connection method you intend to use (we'll see more on this later)-you must make an ActiveSync connection between the device and the PC running Studio. For more detailed steps on making an ActiveSync connection and persisting partnership information, see the ADS Developer's Getting Started Guide for CE.

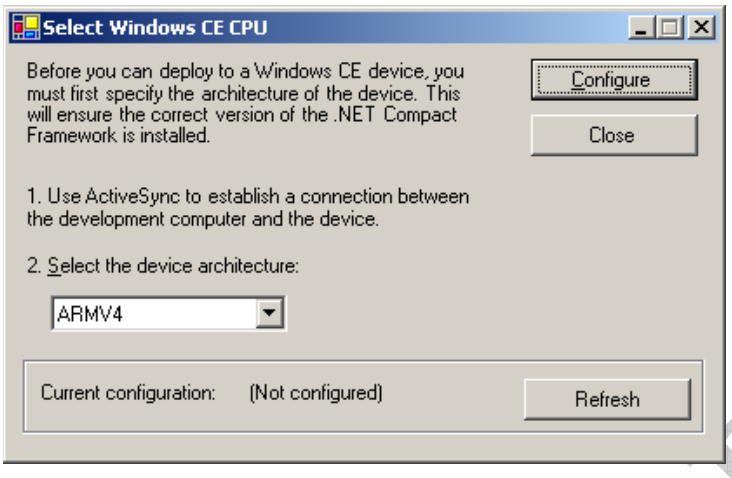
3.2.2 Step 2 – Select your Device CPU

Next you must tell Visual Studio which CPU type the device you're using has. The reason this must be manually done is that ActiveSync currently does not expose any method of determining this automatically.

To configure Studio for your device processor, use the *Select Windows CE Device CPU* Tool which is available under the *Tools* menu on Studio. If this option is not in your *Tools* menu you must install the Visual Studio 2003 Add-On pack (see Section 1).

The *Select Windows CE CPU* tool is a basic, single dialog utility with a dropdown list box and a couple of buttons.

PRELIMINARY



Once you've made an ActiveSync connection to your device, select the CPU architecture on your ADS device from the dropdown list. If your device is based on the Intel SA1110 StrongARM (like the Bitsy Plus, Graphics Master or Graphics Client Plus) then select **ARMV4**. If your device is based on the Intel PXA255 XScale (like the BitsyX or AGX) then select **ARM4T**.

3.2.3 Step 3 – Making the Connection

At this point the steps for connecting to the device diverge depending on the method of connection transport. The method of transport is largely determined by your physical connection to the device:

- If you have made an ActiveSync connection via RS-232 serial or USB then you *must* use the Connection Manager Client.
- If you have made an Ethernet ActiveSync connection then you have the option of using either the Connection Manager Client or the Smart Device Authentication Utility.

The ideal connection method for both ease of use and speed of debugging is to use the Smart Device Authentication Utility over Ethernet. If it is at all possible, this is what we recommend you use. We will, however, cover both methods of connecting.

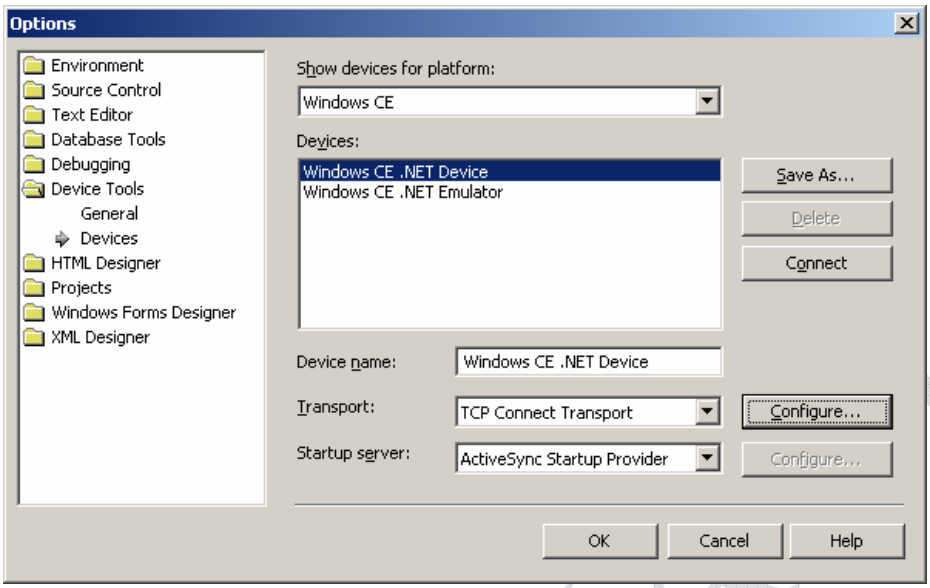
3.2.3.1 *Using the Connection Manager Client*

If you have a RS-232 serial or USB ActiveSync connection to your device, your only option for connectivity is to use the Connection Manager Client, which is comprised of an executable (ConManClinet.exe) and a Transport Library (Cmtnpt_TcpConnect.dll) that must be in the *\Windows* folder on your CE device.

Visual Studio may or may not automatically copy the necessary files to your device. It is simplest to always see if Studio will push the files automatically before doing so manually.

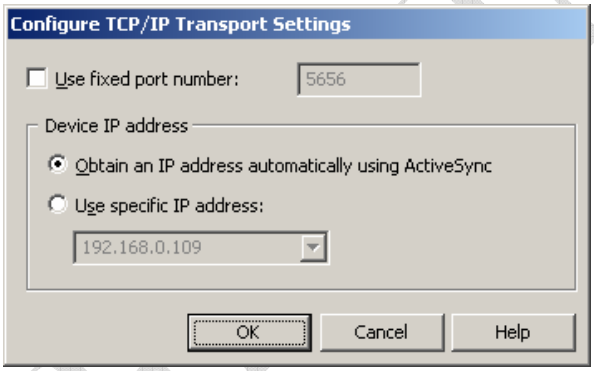
First, you must set up your device options. From Studio's *Tools* menu, select *Options*. You will be presented with the Visual Studio Options dialog. In the left pane of the dialog, open the *Device Tools* folder and select the *Devices* option. You will see the following dialog:

PRELIMINARY



Choose the "Windows CE" platform from the dropdown list and then select "Windows CE .NET Device" from the list of devices presented.

Next make sure that the Transport selected is "TCP Connect Transport" and then click *Configure*. You will get another popup dialog:



On this dialog make sure that "Use fixed port number" is **not** checked and that "Obtain an IP address automatically using ActiveSync" **is** selected, then click *OK* to return to the Options dialog.

Back on the Options dialog, make sure that the Startup Server is set to "ActiveSync Startup Provider", then click on the *Connect* button.

At this point Studio will attempt to connect to the device and will provide progress information in the status bar area at the bottom of the main Visual Studio form as well as adding progress information to the Output window if it is visible.

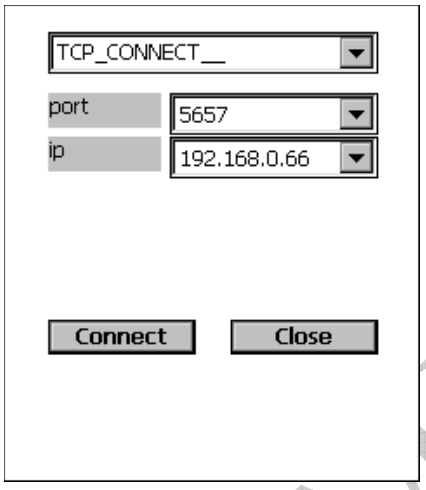
It is during this process that Studio should copy the Connection Manager Client files to your device. If it does not, you will get an error dialog after approximately 30-60 seconds like this:

PRELIMINARY

If Studio fails to automatically connect and copy the files for you, you will have to manually copy them. On your development PC find the following files: `ConManClient.exe`, `Cmtnpt_TcpConnect.dll`. If you used the default installation path for Visual Studio then they will be found in `C:\Program Files\Microsoft Visual Studio .NET 2003.0\CompactFrameworkSDK\ConnectionManager\Target\wce400` under the folder corresponding to your device processor (again ARM4 for the SA1110 or ARM4t for the PXA255).

Copy the two files to your device's *\Windows* folder in whatever manner you choose (via ATA card, Windows Explorer, the Remote File Viewer, etc.).

Now, on the device, run `ConManClient.exe` and you will get the following dialog:



Make sure that the top dropdown list is set to "TCP_CONNECT__" then enter the IP address of the PC where you are running Visual Studio in the ip dropdown. Leave the *port* dropdown to its default value.

Now tap the *Connect* button on the device and again click the *Connect* button in the Options dialog of Visual Studio. After a few seconds the status bar in the main form of Studio should read "Device Connected". If it does not, move on to the Troubleshooting your Connection section later in this chapter.

You are now set up for debugging.

3.2.3.2 *Using the Smart Device Authentication Utility*

The fastest method for debugging is using Microsoft's Smart Device Authentication tool with an Ethernet connection. The Smart Device Authentication tool is a two-part utility that comes with the Visual Studio 2003 Add-on Pack. The device-side piece may or may not be pre-installed in the ADS CE image you are using.

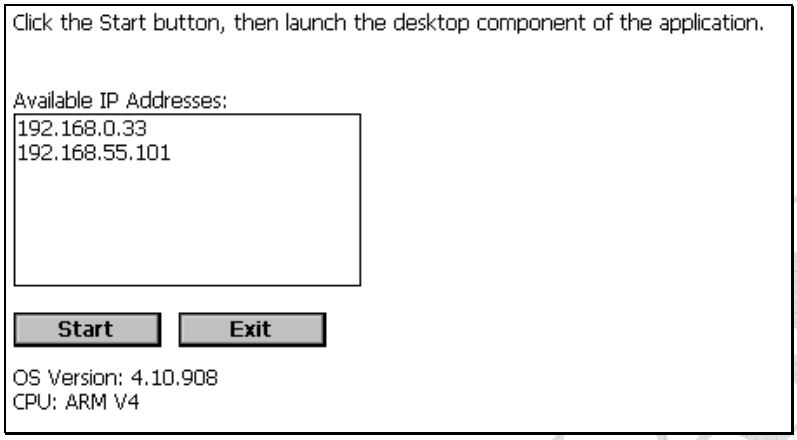
The device-side piece is `SDAuthUtilDevice.exe`. First check in the *\Windows* folder of your CE device to see if it has been included in the image you are using, making sure that you have selected to view all files through the *View | Options* menu item of Explorer. If `SDAuthUtilDevice.exe` is not in your *\Windows* folder, manually copy it there from your development PC.

If you used the default installation path for Visual Studio it can be found in the *C:\Program Files\Microsoft Visual Studio .NET 2003\CompactFrameworkSDK\WinCE Utilities\Authentication Util\WinCE4* folder under the subfolder corresponding to your device

PRELIMINARY

processor (ARM4 for the SA1110 or ARM4t for the PXA255). Again, if you cannot find this folder on your PC, you must install the Visual Studio 2003 Add-on Pack.

Once you have `SDAuthUtilDevice.exe` in the *\Windows* folder on your device, go ahead and run it. You will see the following dialog:



From this dialog select your device's IP address but **do not** tap any buttons yet. Note that in this example two IP addresses are presented. 192.168.55.101 is an ActiveSync-assigned IP address and is **not** usable by the Smart Device Authentication Utility. You must use the IP address assigned to the device's network card.

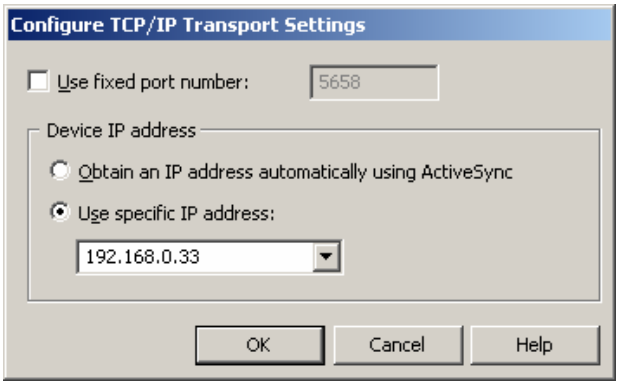
Now, on your development PC under Visual Studio's *Tools* menu, select "Smart Device Authentication Utility". This will bring up the following application dialog:

In the "Device IP address" text box, enter the IP address that you selected in the device dialog in the previous step. Now tap the *Start* button on the device, wait a second, then click *Set Up Device* on the PC. At this point the PC will find the device and present the following dialog:

Now that the device and your PC have negotiated a connection, you must tell Studio to use this device for debugging. Back on the PC, if it is not still open, open up the Devices Option dialog again. Make sure that "TCP Connect Transport" is selected in the *Transport* dropdown list and

PRELIMINARY

click *Configure*. In the configuration dialog select the "Use Specific IP Address" radio button and enter the device's IP address in the text box:



Now click OK to close the Configure dialog, then click the *Connect* button in the Options dialog of Visual Studio. After a few seconds the status bar in the main form of Studio should read "Device Connected". If it does not, move on to the Troubleshooting your Connection section later in this chapter.

You are now set up for debugging.

3.3 Troubleshooting your Connection

If you are still unable to make a connection to your ADS device to enable debugging, there are some steps you can take to try to remedy the situation.

WARNING

To keep a solid connectivity between Studio 2003 and your ADS device it is important to prevent the information Studio uses for the connection from being corrupted. Studio is constantly transferring data to and from the device and interrupting the communications in a way that Studio is not expecting can cause problems so remember to *always stop the debug session* before resetting or removing power from the device or removing the communication cables.

If your device loses power or is reset while an active debug session is running, you will most likely have to repeat the device connection steps above after deleting the existing information (the steps are outlined below).

3.3.1 Check your Software Installation Order

You must have installed your development software in the following order:

- ActiveSync
- Visual Studio
- Visual Studio Add-on Pack

If you did not follow this order, you will not be able to connect to your device (the ActiveSync install will overwrite necessary Studio registry keys).

PRELIMINARY

3.3.1.1 *Hard-Reset your ADS Device*

The first step in trouble shooting connectivity problems is to hard-reset your device. This is best done by removing all power to your device, including any backup batteries if you have them. If your device has supercaps installed, make sure that they have also been fully discharged.

3.3.1.2 *Run `DelCryptoKeys.exe`*

The first step in troubleshooting your connection is to ensure there are no orphaned Studio communication encryption keys on the device.

Visual Studio uses encryption to pass debug information to and from the Windows CE device to protect the device from unwanted intrusion. Unfortunately, these encryption keys sometimes get orphaned and the device will stop responding to Studio, assuming it is an unauthorized contact.

The Visual Studio 2003 add-on pack ships with a device tool that will delete the crypto keys used by Studio and your ADS device may or may not have it already included in your CE image. The tool name is `DelCryptoKeys.exe`. If you cannot find it in the *\Windows* folder of your device, you can manually copy it to your device.

If you installed Studio to its default installation path, you will find the utility in the *C:\Program Files\Microsoft Visual Studio .NET 2003\CompactFrameworkSDK\WinCE Utilities\DelCryptoKey* folder of your PC under the subfolder corresponding to your device processor (ARM4 for the SA1110 or ARM4t for the PXA255).

Just run `DelCryptoKeys.exe` and retry the steps for connecting to your device.

3.3.1.3 *Disconnect and Reconnect the Smart Device Authentication Utility*

If you're using the Smart Device Authentication utility, disconnect and reconnect it.

3.3.1.4 *Delete your Persistent Registry*

If running `DelCryptoKeys.exe` does not correct your connection problems, the next step is to erase your device's persistent registry, if it has one. See the Developer's Getting Started Guide for CE for more information on determining if you have a persistent registry and how to erase it.

3.3.1.5 *Reboot your PC*

Rebooting your PC will restore a base state for your system hardware and software.

3.3.1.6 *Contact ADS Technical Support*

If all other steps have failed, try contacting ADS technical support at support@applieddata.net. We may be able to provide insight into the problem or recommend further steps or resources.

3.4 Debugging

Visual Studio 2003 supports robust application debugging directly on your ADS device. You can set breakpoints, step through your source code line by line, view variables in real time, walk the call stack and other productivity enhancing tasks. We will briefly cover some of them here. For more information, we highly recommend reading the documentation that comes with Visual Studio.

Breakpoints

{Section To be completed}

PRELIMINARY

{screen shot}

3.4.1 Watches

{Section To be completed}

{screen shot}

3.4.2 Debug Output

Unlike desktop projects, Smart Device projects are unable to send custom data back to the Visual Studio IDE during debugging. This means that you cannot call {method name} in your code and get output in Studio's Debug window. If you want to log progress or events from your code or provide a application trace output you must use another method.

Console.WriteLine

3.4.2.1 *Log File*

3.4.2.2 *Debug Port*

P/Invoke NkDbgPrintf

{insert sample code}

```
Template for code sections
```

4 Installing and Running Smart Device Applications

.NET Smart Device applications require the .NET Compact Framework (CF) runtime files be installed on the device. The CF runtimes are installed in the Global Assembly Cache (GAC) and as such must reside in the \Windows folder.

Some ADS Windows CE builds come with the Compact Framework already included in the image. You can manually check to see if the runtimes are in your image by checking the \Windows folder for eleven files all beginning with "gac_" (e.g. gac_system.windows.forms_v1_0_5000_0_cneutral.dll).

NOTE

Some early ADS builds had the CF runtime files included but were missing the necessary registry entries for the GAC installation. This omission renders the installed CF runtimes unusable and the CF runtimes must be reinstalled on these devices for Smart Device applications to run.

4.1 Installing the .NET Compact Framework Runtimes

If your system does not have the CF runtimes installed or you need to install a newer version of the runtimes, follow these steps:

1. **Copy the Compact Framework Redistributable CAB file to your device.**

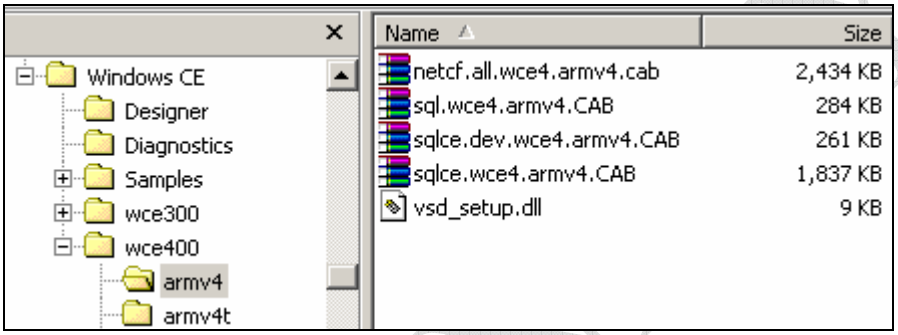
   All of the CF redistributable CABs are installed on your development PC when you install Visual Studio 2003. A default installation will place them in the following folder:

PRELIMINARY

```
C:\Program Files\Microsoft Visual Studio .NET
2003\CompactFrameworkSDK\v1.0.5000\Windows CE\wce400
```

The CF runtimes are processor dependent therefore this folder will contain several subfolders, one folder for each processor architecture supported by the CF. ADS systems use either the ARMV4 (for the StrongARM processor) or the ARMV4T (for the XScale processor) architecture.

Inside the subfolder for your processor you will find four CAB files. One is the redistributable for the CF and the remaining three are for SQL Server CE. In the example shown below, netcf.all.wce4.arm4.cab is the Compact Framework redistributable.



Copy the redistributable file to your development system. The target location is not important, as the file will be extracted to the correct location when run in the next step.

2. **Run/Extract the Redistributable**

Extract the redistributable file by double-tapping it or by running wceload.exe with the fully qualified name of the CAB file as a command-line parameter.

You will be prompted for a installation directory during the install process. Because the CF runtimes must be installed in the GAC, it is important that you do not change this from the default.

NOTE

If your ADS system has a persistent registry, the act of running a CAB file will automatically call RegFlushKey() and persist your current registry, including the installation information for the Compact Framework.

If you installed the CF runtimes on a device that did not have them in the image, or if you install a newer version of the CF runtimes, the binaries will be erased if the device is reset or loses power, but the installation information will still be resident in the device registry. This will cause an "application is already installed" warning when you reinstall the CF runtimes.

TIP

Wceload.exe will automatically delete any CAB file it runs after installation. If you want prevent the deletion, mark the file as read-only.

PRELIMINARY

4.2 *Installing Your Smart Device Application*

Installing your Smart Device application can be done in two ways:

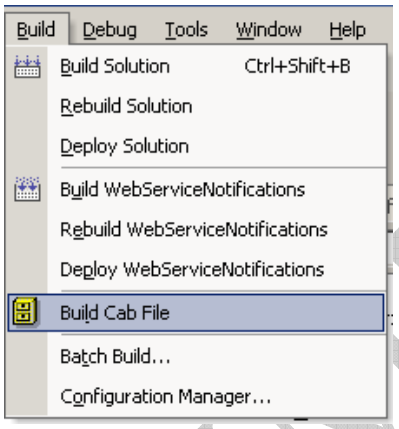
1. **Copy the target binaries to the device**

   For most applications with only a few binaries, it is simplest to just copy your binaries to the target device. You can place the binaries in any folder, but if you're using custom non-GAC DLLs, they must reside in the same folder as your application.

2. **Build and install a redistributable CAB**

   If your application is more complex, needs to install assemblies in the GAC or you want to have uninstall information saved to the device, then you must use a redistributable CAB file to deploy your application.

   Redistributable CABs are built by selecting "Build Cab File" from the *Build* menu item in Visual Studio (which in turn invokes CABWIZ.EXE).



This will generate separate CAB files for each processor/platform combination possible, all of which will end up in a subfolder inside you Visual Studio project's folder named "cab". The procedure for installing the generated redistributable CAB file is the same as for installing the CF runtimes outlined earlier.

TIP

The "Build Cab File" menu item simply invokes CABWIZ.EXE. There are several shareware and commercial applications available that wrap CABWIZ.EXE, plus decent documentation available online which provides the ability to customize the CAB file creation if you so desire.

5 Remote Tools

Below we outline several tools that are very useful for helping debug and document applications that do not come with Visual Studio 2003.

5.1 Adding External Tools to Studio

{screen shots}

5.2 Remote Registry Editor

From eVC

{Section To be completed}

PRELIMINARY

5.3 Remote Zoom-In

From eVC

{Section To be completed}

5.4 CE Remote Display PowerToy

http://www.microsoft.com/windowsmobile/resources/downloads/pocketpc/powertoys.mspx

{Section To be completed}

5.5 Remote File Viewer

From eVC

{Section To be completed}

6 Adapting the System to your Needs

{Section To be completed}

6.1 *Third-Party Drivers and CAB files*

{Section To be completed}

6.2 *Persisting Applications and Settings*

6.2.1 ADSCopy

{Section To be completed}

6.2.2 ADSReg

{Section To be completed}

6.2.3 ADSAutorun

{Section To be completed}

6.2.4 The Persistent Registry

{Section To be completed}

6.3 *Customizing the Device Environment*

6.3.1 Backlight Settings

{Section To be completed}

6.3.2 Sleep Settings

{Section To be completed}

PRELIMINARY

6.3.3 Registry Settings

{Section To be completed}

6.3.4 Restarting Drivers

{Section To be completed}

6.4 Optimizing Performance

{Section To be completed}

6.4.1 Threads

Caution
Caution template

{Section To be completed}

6.4.2 Intel GPP/IPP

{Section To be completed}

6.4.3 System Tradeoffs

{Section To be completed}

6.4.4 Video, Images and Display Controllers

{Section To be completed}

6.5 Preparing for Production

{Section To be completed}

6.5.1 Startup Folders

{Section To be completed}

6.5.2 Suppressing the Desktop

{Section To be completed}

6.5.3 “Headless” Devices

{Section To be completed}

7 System Driver Reference

{Section To be completed}

PRELIMINARY

7.1 FlashFX Disk

{Section To be completed}

7.2 Input Devices

{Section To be completed}

7.3 Ethernet

{Section To be completed}

7.4 Serial Communication

{Section To be completed}

7.5 Device I/O

{Section To be completed}

7.6 Interrupts

{Section To be completed}

8 Understanding the Boot Process

{Section to be completed}

8.1 RAM Usage

ADS Devices do not use Execute In Place (XIP) for any part of the Windows CE image. This has important implications on the amount of RAM that will be available to applications running on the system.

Every time an ADS system boots, the Windows CE image is copied from on-board flash memory into RAM and then is executed from RAM. The result of this boot model is that the RAM used to hold the image is not usable or even seen by the OS itself as usable memory.

Once the image is copied to RAM and Windows CE starts up, another significant chunk of RAM is grabbed by the OS. This RAM is for heap and stack usage by the CE kernel, the CE device manager, drivers and any core applications.

All of the previously mentioned RAM allocation takes place before the ADS device is fully running and cannot be altered. The remaining RAM is available for two possible uses by applications on the device: Storage Memory and/or Program Memory.

Storage Memory is RAM usable by the CE file system and is analogous to hard drive space on a PC. Any files stored in the device's file system takes up Storage Memory, with exceptions for persistent or removable storage. This means that files in the "FlashFX Disk" or "Storage Card" folders do not require Storage Memory, but any other files, including data files created by applications, require Storage Memory.

Program Memory is RAM that can be allocated and used by applications for their heaps and/or stacks and is analogous to the RAM on a PC.

The division between Storage Memory and Program memory can be adjusted for the current session using the System Memory Control Panel applet, permanently by modifying the device

PRELIMINARY

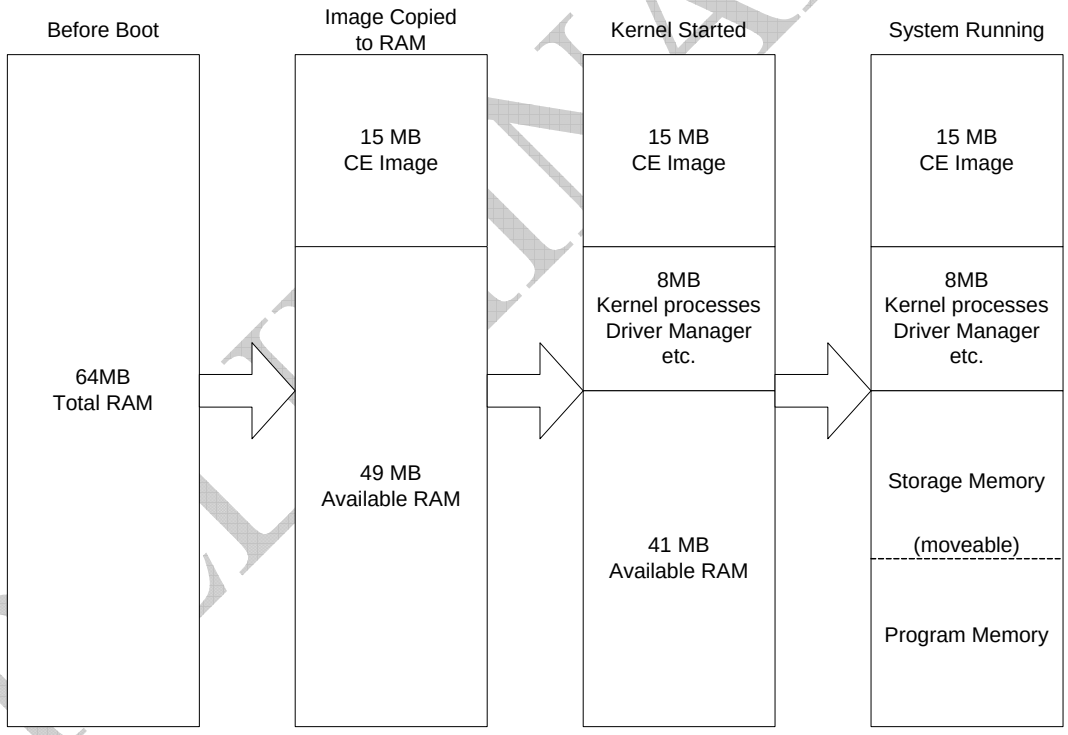
registry (see the Developer's Getting Started Guide), or dynamically at run time using the `SetSystemMemoryDivision`() API.

Let's look at an example, which is also diagrammed below. Assume you have a BitsyX development system with the standard 64MB of SDRAM and you are using a 15MB Windows CE image. When the system boots, the image is copied to RAM, leaving 49 MB. Next the kernel, device manager, device servers and drivers takes up about another 8 MB of RAM.

This leaves about 41 MB of RAM for your application. If your application requires a lot of memory, for instance an intensive multimedia application, then you may want to allocate more Program Memory for your application to have available. If you application stores large amounts of data into a database file then you may want to allocate more Storage Memory for the application to use.

As a general guideline, starting with a fairly equal division of memory between Storage and Program Memory will work initially. The actual division can then be either modified based on data from testing or your application could implement its own RAM manager to dynamically adjust the division as necessary.

ADS Device RAM Usage - Not all RAM on the system will be available for application usage.

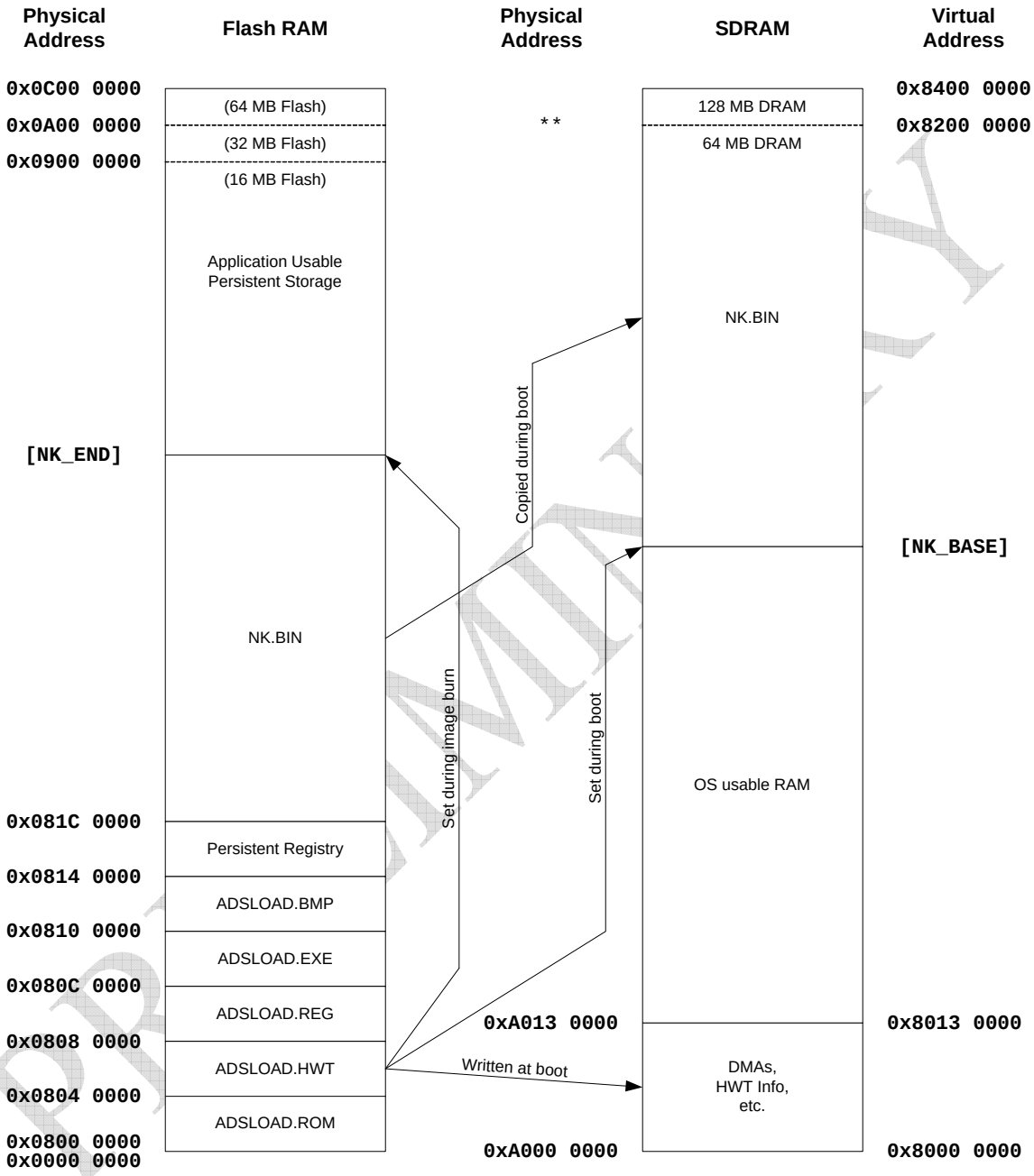


8.2 *CE Memory Maps*

{Section to be completed}

PRELIMINARY

ADS BitsyX Memory Map



PRELIMINARY