Installation and User's Guide

ConnectCore XP 270
Board Support Package
for Windows CE 5.0

Making DEVICE NETWORKING easy

Digi International Inc.
11001 Bren Road East
Minnetonka, MN 55343 (USA)
☎ +1 877 912-3444 or +1 952 912-3444
http://www.digi.com

FS Forth-Systeme GmbH
Kueferstrasse 8
79206 Breisach (Germany)
☎ +49 7667 908-0
http://www.fsforth.de

Sistemas Embebidos S.A.
Calvo Sotelo 1, 1º - Dcha
26003 Logroño (Spain)
☎ +34 941 270 060
http://www.embebidos.com

Table of Contents

Documentation History

Date	Version	Author	Description
12/12/2005	1.0		Release Version
28/11/2005	0.10	Carlos Marín	Revision
25/11/2005	0.9	Mike Engel Héctor Palacios Peggy Sanchez	Revision
09/11/2005	0.8	Héctor Palacios	Integrated feedback from developers and testers
04/11/2005	0.7	Héctor Palacios	Full revision. Formatted to new template format
27/10/2005	0.6	Mike Engel	Added several new chapters
20/10/2005	0.5	Mike Engel	Updated installation process
11/10/2005	0.4	Mike Engel	Updated U-boot chapter
06/10/2005	0.3	Héctor Bujanda	Added GPIO driver, testDRV & SDK
30/09/2005	0.2	Mike Engel	Updated with new features
02/09/2005	0.1	Mike Engel	Initial version

1. Introduction

1.1. Overview

The Board Support Package for the ConnectCore XP family for Microsoft Windows CE 5.0 contains all the necessary software components to allow a simple and fast start-up of application development with the Windows CE 5.0 hardware platform.

The Board Support Package reduces the time to market phase for software leveraging Microsoft Windows CE 5.0 running on the ConnectCore XP (CCXP) module. With the Ethernet controller SMSC91C111 on the CCXP, you can now download and debug your platform without additional network cards.

1.2. License background

The BSP includes the full source code for the boot loader as well as of all drivers. The source code is intended to be used internally for development and debugging only. Distribution of the original or modified source code is forbidden.

There are no royalties for the Windows CE images created with the BSP running on Digi hardware. However, there are royalties for all images running on non-Digi hardware. Royalties accrue for boot loaders and drivers.

The Datalight flash file system, *Flash FX Pro,* must be licensed separately. For every target runtime licenses must be purchased. The source code of *FlashFX* is also available on an individual basis for an additional fee.

For detailed information about Licensing and Royalties please contact your sales representative.

1.3. Features

CCXP Board Support Package for Windows CE 5.0

- LCD display driver
- Ethernet driver
- USB Host/Device driver
- Touch screen driver
- FlashFX® driver
- PCCARD driver
- IP2REG driver
- GPIO driver
- Hive based registry

Bootloader

- U-Boot version 1.1.3 with splash screen support
- Reduced EBOOT version to load standard Windows CE kernel from U-Boot

Toolchain

- Microcross GNU X-Tools™
- U-Boot source code

1.4. Conventions used in this manual

The following is a list of the typographical conventions used in this manual:

Style	Used for file and directory names, programs and command names, command-line options, URL, and new terms.
`Style`	Used in examples to show the contents of files, the output from commands or in the text the C code.
`Style`	Used in examples to show the text that should be typed literally by the user.
#	Used to indicate the listed commands have to be executed as root.
$	Used to indicate the listed commands have to be executed as a normal user.
[1]	Used to reference an item of the reference section.

This manual also uses these frames and symbols:

This is a warning. It helps you to solve or to avoid common mistakes or problems

This is a tip. It contains useful information about a topic

```
$   This is a host computer session
$   And this is what you must input (in bold)
```

```
#   This is a target session
#   And this is what you must input (in bold)
```

1.5. Acronyms and abbreviations

BSP	Board Support Package
CCXP	ConnectCore XP 270
GPIO	General Purpose Input/Output
HID	Human Interface Device
IOCTL	I/O control
IRQ	Interrupt Request
JTAG	Joint Test Action Group (IEEE 1149.1)
LCD	Liquid Crystal Display
MBR	Master Boot Record
OAL	OEM Abstraction Layer

OEM	Original Equipment Manufacturer
QFE	Quick Fix Engineering
SDK	Software Development Kit
TFTP	Trivial File Transfer Protocol
U-Boot	Universal boot loader
USB	Universal Serial Bus

2. Requirements

2.1. System requirements

To develop with CCXP BSP, your development workstation has to meet the following requirements:

- x86 PC with 500 MHz Pentium III or faster processor; 2 GHz Pentium 4 or equivalent recommended
- Microsoft Windows 2000 Professional with Service Pack 4 or Windows XP Professional with Service Pack 1.
- 256 MB of RAM; 512MB recommended
- 300 MB of available hard-disk space for installation of the BSP and the provided SDK.
- CD-ROM or DVD-ROM drive.
- Serial port
- Parallel port (only for JTAG flashing)
- Ethernet network card
- Microsoft Windows CE 5.0.
- Microsoft .NET Framework version 1.1.
- TFTP server

For best results, we recommend installing one of the following development tools for application development:

- Microsoft Embedded Visual Tools 4.0.
- Microsoft Visual Studio .NET.

2.2. TFTP server

U-Boot is capable of writing files to the Flash memory of the module. A TFTP server is required to transport these files from your host computer to the target.

If you don't own a TFTP server you can download SolarWinds free TFTP server from *http://www.solarwinds.net/Tools/Free_tools/TFTP_Server/*

2.2.1. Configuring SolarWinds TFTP server

If you installed SolarWinds TFTP server, start the application and select *File* > *Configure* from the menu. The following window will pop up:

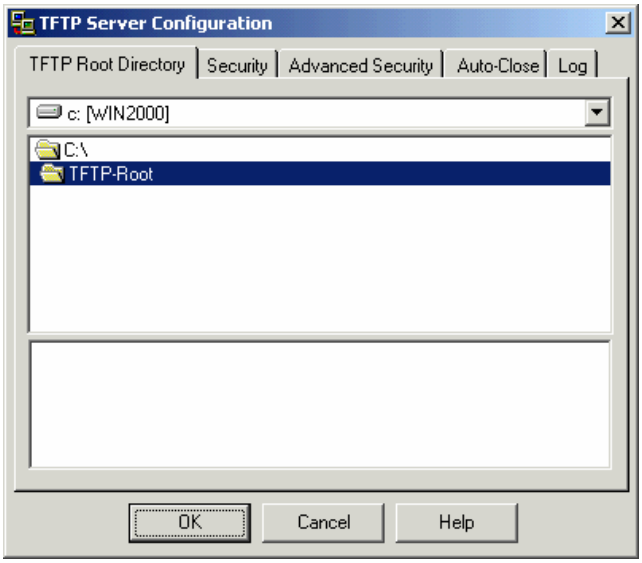


Figure 2-1: Configuration of SolarWinds TFTP server (Root directory)

Select the drive and folder that you will expose to the TFTP clients. Go to the Security tab and select *Transmit and Receive files*:

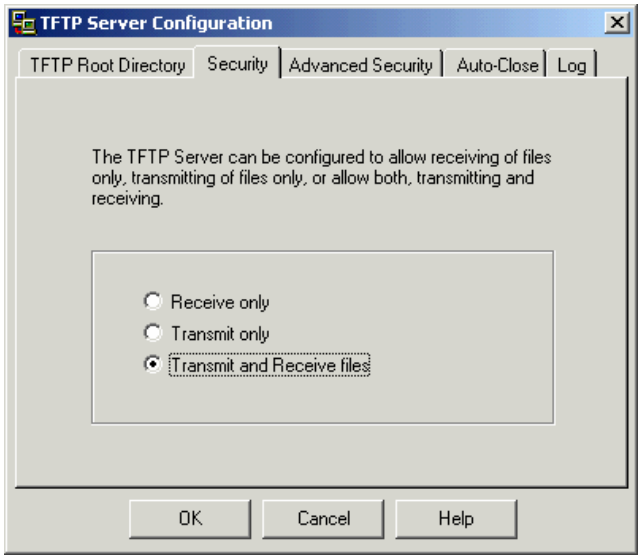


Figure 2-2: Configuration of SolarWinds TFTP server (Security)

Click OK to close the Configuration window.

2.3. JTAG-Booster

The JTAG-Booster software for the hardware Flash update is a DOS application. To install the JTAG-Booster software, copy the directory "hardware" from the CD to any directory on the hard disk. This directory may also contain a file “Readme.txt” with latest instructions.

Ensure the parallel port is accessible for applications. Install the "Kithara DOS Enabler" which is shipped on the CD.

A detailed manual can also be found on the CD in the folder "hardware".

3. Getting Started

3.1. Hardware setup

This chapter describes how to configure and test your host PC, the development board with the module (target) and how to start up the target for the first time.

Step 1: Connect serial port

Connect COM1 of the host PC to Serial Port 1 on the development board (target) using a standard serial modem cable, included in the kit. The serial connection is used to communicate with the target device.

Step 2: Connect Ethernet interface

Establish the Ethernet connection by connecting an Ethernet crossover cable directly to the development board's Ethernet port and your host PC. Alternatively, if you already have a running network configuration, you can connect the development board to your hub or switch.

Step 3: Configure HyperTerminal

Configure HyperTerminal to view the console output the target prints on the serial interface.

To configure HyperTerminal, go to *Start* > *Programs* > *Accessories* > *Communications* > *HyperTerminal*

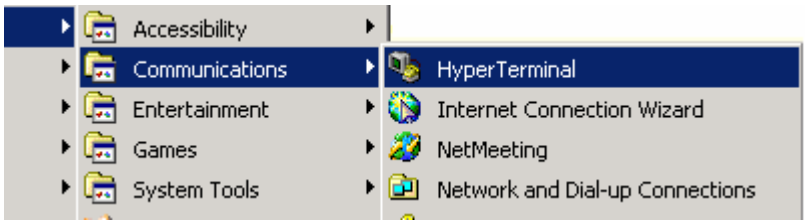


Figure 3-1: HyperTerminal

Enter the name of the session and click OK.

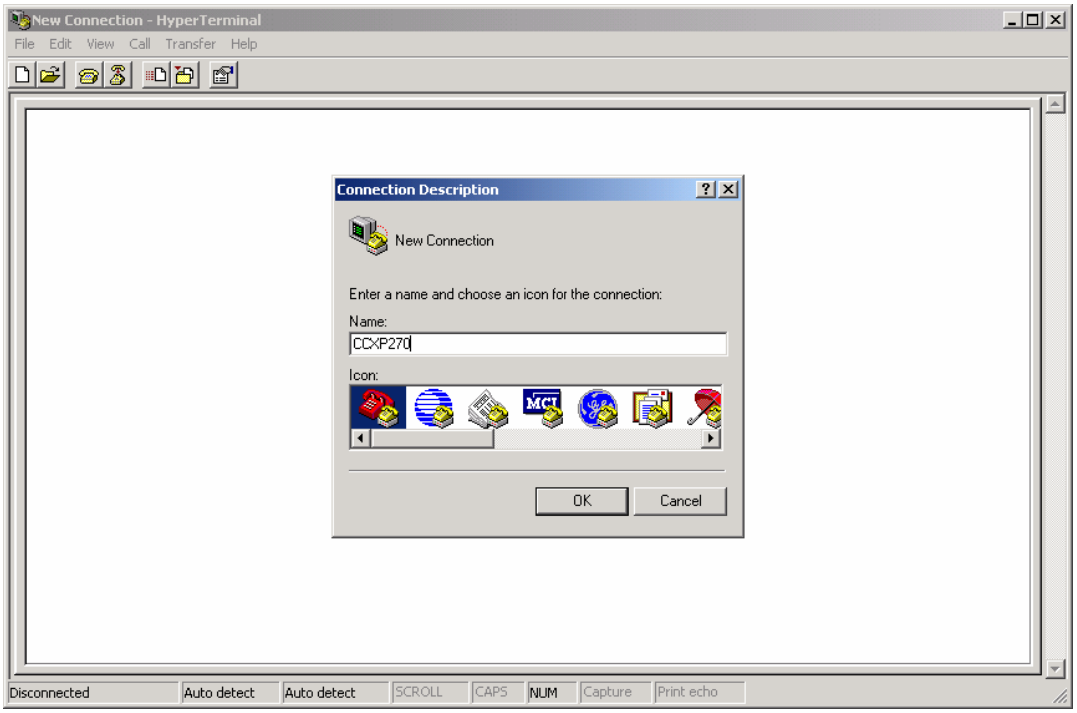


Figure 3-2: HyperTerminal (connection description)

Select the host PC's serial port (the one you connected the serial cable to) :

Figure 3-3: HyperTerminal (serial port)

Finally, configure the serial parameters for 38400 bits per second, no parity, 8 data bits, 1 stop bit and no flow control.

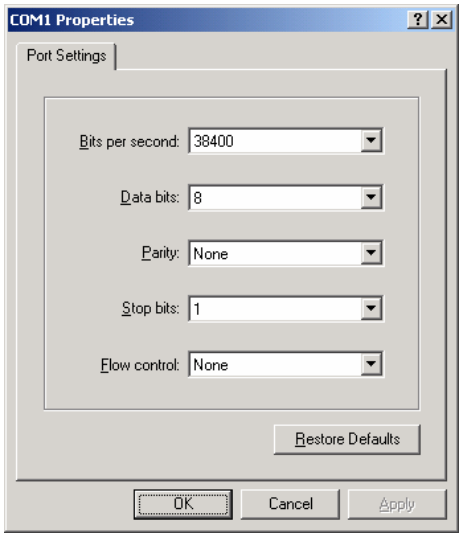


Figure 3-4: HyperTerminal (serial parameters)

Step 4: Connect power

Connect the power supply to the development board. As the board powers up, the LEDs will light. Approximately 2-4 seconds later the system will print boot messages on the console.

After 30-35 seconds, when the boot loader has unpacked and launched the pre-installed Windows CE kernel from the module's built-in Flash memory, you will see output on the terminal client similar to the following:

```
U-Boot 1.1.3 (Nov  9 2005 - 10:02:07)  FS.2
for FS Forth-Systeme CCXP270 on STK2

U-Boot code: A0090000 -> A00B2F74  BSS: -> A00F4324
RAM Configuration:
Bank #0: a0000000 64 MB
Bank #1: a4000000  0 kB
Bank #2: a8000000  0 kB
Bank #3: ac000000  0 kB
Flash: 32 MB
In:    serial
Out:   serial
Err:   serial

Detecting SMSC LAN91C111 Ethernet Controller...
Revision ID: 0x01
Chip ID: 0x09
SMSC LAN91C111 detected...
Get MAC Address from 1-wire EEPROM
MAC Address from EEPROM : 00:04:F3:00:38:10
Initializing ethernet...
Using MAC Address 00:04:F3:00:38:10
Hit any key to stop autoboot:  0
## Starting application at 0xA0100000 ...
```

Step 5: Test Ethernet configuration

The target uses a default IP address on the 192.168.42.x network. We recommend that you configure a dedicated Windows CE development network separate from your standard network. Simply add and configure an additional network interface card to your PC and use an IP address from the 192.168.42.0 subnet, e.g. 192.168.42.1.

The target network parameters can be changed in the U-Boot bootloader using the "setenv" command.

On the target you can see the IP address by going to Windows CE *Start* > *Settings* > *Network and Dial-up Connections*. Double click the LAN91C111_1 network interface to configure the network parameters:

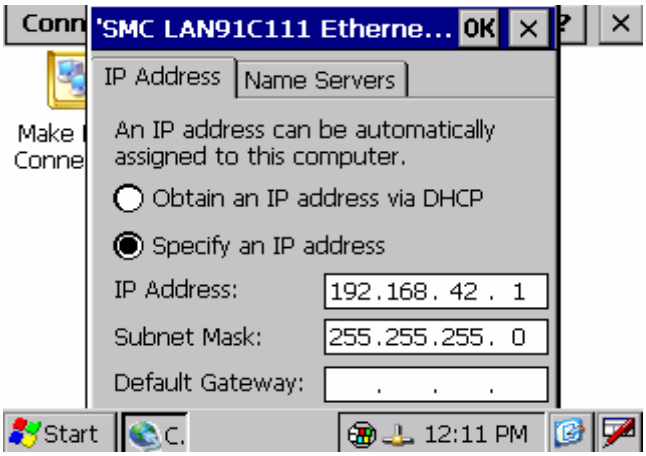


Figure 3-5: Network configuration dialog

3.2. Software installation

First you have to install Microsoft Platform Builder 5.0 from the original Microsoft Windows CE 5.0 DVD/CD. If you use Microsoft Windows 2000, note the following:

Before you install Platform Builder under Windows 2000 you must install Windows 2000 Service Pack 4 and the .NET Framework.

During the installation of Platform Builder you must select ARMV4I and XSCALE as CPU support; without it the BSP installation will fail.

If you are using Windows 2000 and have Internet Explorer 5.0 or earlier, you may receive a Java error message when opening Platform Builder. Updating Internet Explorer to the latest version will fix this issue.

Once you have installed Platform Builder, start the *Setup.exe* on the CD of the BSP and follow the instructions. A wizard will guide you through the installation process. The wizard will install the BSP, QFEs, and SDK by default; however, you can install additional files like the *Cygwin* tool chain or the U-Boot bootloader sources.

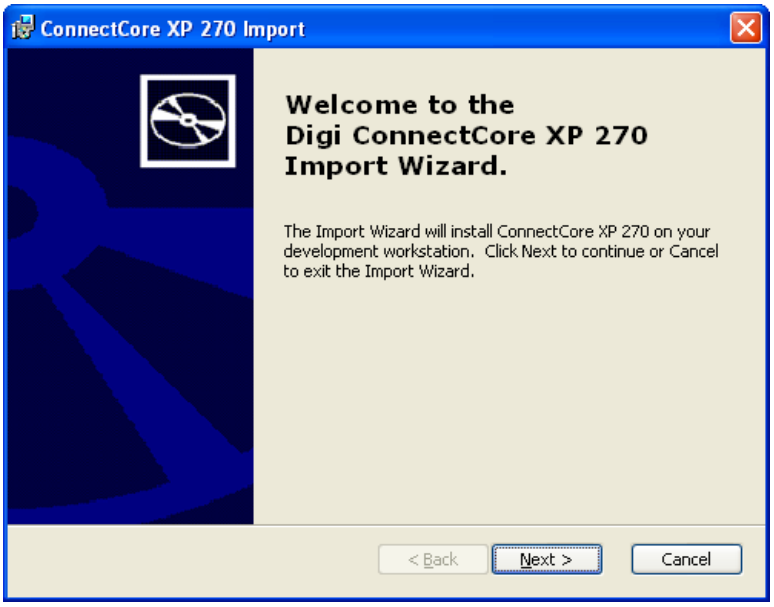


Figure 3-6: BSP Installation

The installation of the BSP will copy the following files and folders to your hard drive:

File/Folder	Path	Description
CCXP.CEC	%_WINCEROOT%\public\common\oak\catalog\cec	WinCE 5.0 components file
CCXP	%_WINCEROOT%\Platform	Platform files and drivers
CCXP	%_WINCEROOT%\Platform\Common\Src\ARM\Intel	Platform specific code
Kerneljoin.exe	%_WINCEROOT%\public\common\oak\misc\	Program to generate raw binary file
KernelCopy.bat	%_WINCEROOT%\public\common\oak\misc\	Copies raw binary file into TFTP server folder
bpostmakeimg.bat	%_WINCEROOT%\public\common\oak\misc\	Start to generate and copy raw binary file
ConnectCoreXP.xml	%_WINCEROOT%\public\common\oak\catalog\newplatformwizards\	Example template shown in this documentation

Table 3-1: Files and folders copied during installation

%_WINCEROOT% is an environment variable of your system that stands for the path to your Windows CE 5.0 root directory (Usually C:\WINCE500)

4. Building the First Image

4.1. Creating the platform

From Platform Builder *File* menu, select *New Platform*.

Enter the name of the project you want to create. Select *Next*.

From the available Board Support Packages (BSPs) select the *CCXP* platform and click *Next*.

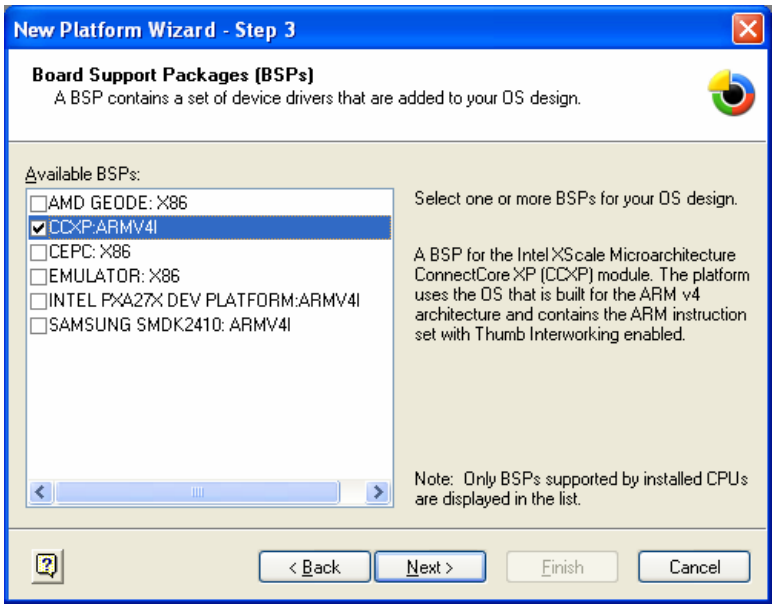


Figure 4-1: Select BSP

Step 4 allows you to select a design template or custom device configuration for your platform. Select *ConnectCore XP 270* to create a standard CCXP based project.

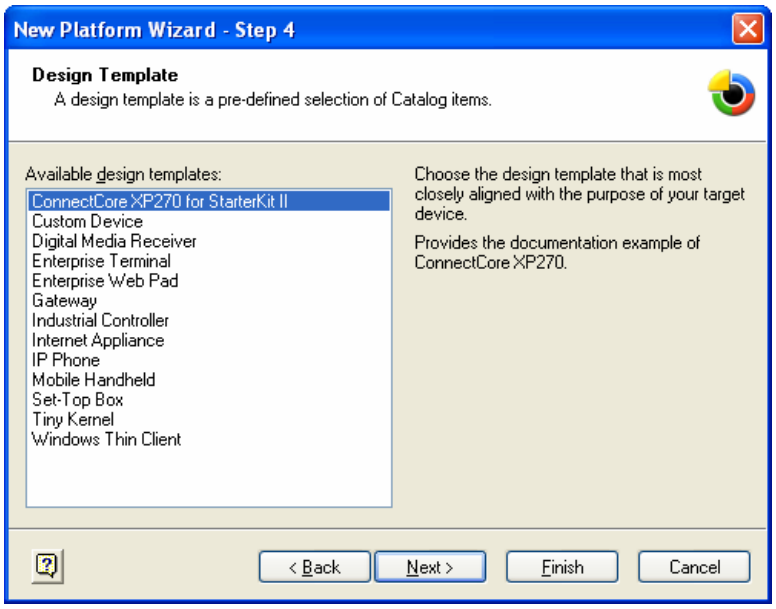


Figure 4-2: Design template

Selecting "Custom Device" lets you choose individual components for your project.

In the following steps (5 -20) you can configure the components and programs that your final Windows CE image will contain. If you have selected the *ConnectCore XP 270* template project, you can either click finish or walk through the configuration step-by-step.

If additional components are needed, Platform Builder will automatically include them in the build.

In this example we will create a kernel for the CCXP development board with support for the hardware. There are two server applications: a Telnet server for remote control of the device and an FTP server for uploading/downloading of files.

Step 5: Applications & Services Development

From the *.NET Compact Framework*, select *OS Dependencies for .NET Compact Framework 1.0* and *.NET Compact Framework 1.0*.

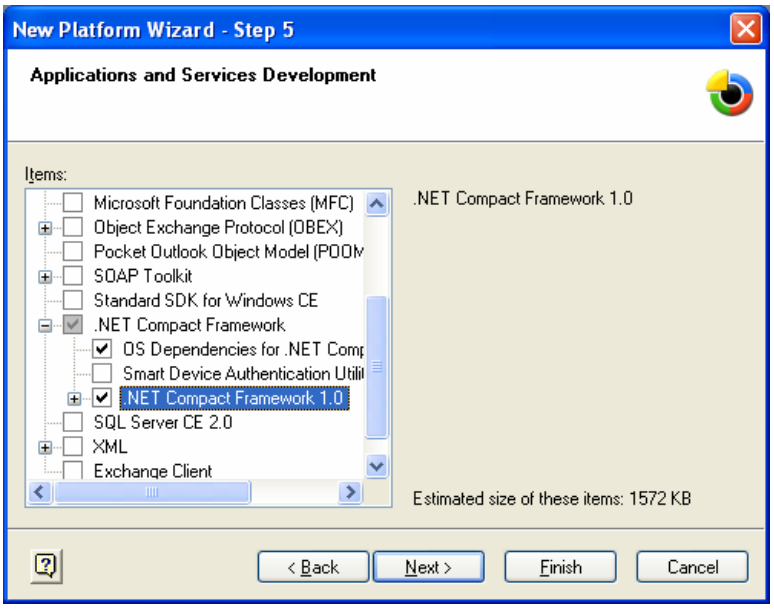


Figure 4-3: Application and Services Development

With .NET Framework, support applications can be written with Visual Studio .NET 2003 for smart devices in Visual Basic or C#.

Step 6: Applications – End User components

For this example we won't need any of these; click *Next*.

Step 7: Core OS Services

Select *USB Host support* (*USB HID Keyboard and* Mouse), *Display Support* and *Serial Port Support,* then click *Next*. If you want to build an image with display support, select the display component. You may also remove it from the project later if needed.

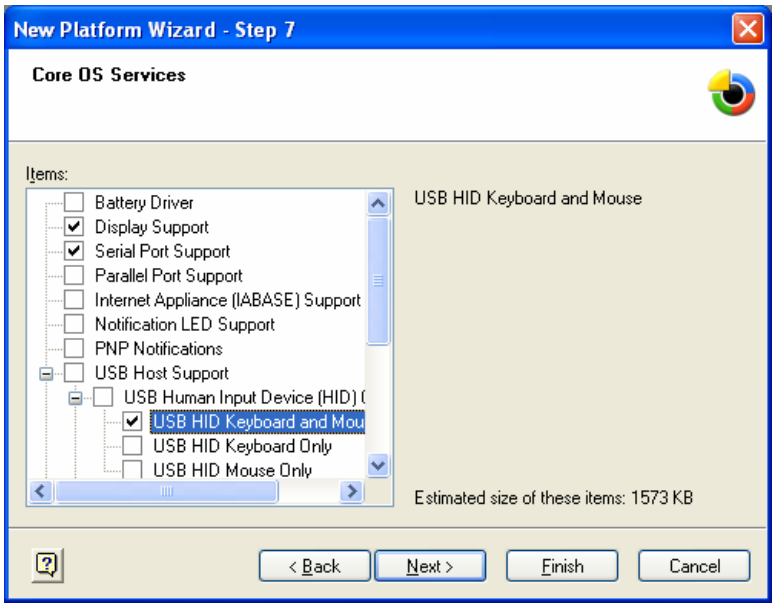


Figure 4-4: Core OS Services

The USB Host, serial port, and Display Windows CE drivers will be included. The registry keys to configure them are stored in the file *%_WINCEROOT%\Platform\CCXP\Files\Platform.reg*

Step 8: Communication services and Networking

Under *Networking Features,* select *Network Utilities (ipconfig, ping, …)* and *Windows Networking API/Redirect*. Under *Networking – Local Area Network (LAN)* you must include the *Wired Local Area Network (802.3, 802.5)* to have support for your Ethernet interface. Under *Servers* select *Telnet Server* and *FTP Server*. With the Telnet Server, you will be able to connect to the Windows CE device shell from anywhere in your network.

To use the internal USB controller, you need to make a hardware modification of the StarterKitII hardware. With this modification you will lose the option to load kernel images from a USB stick.

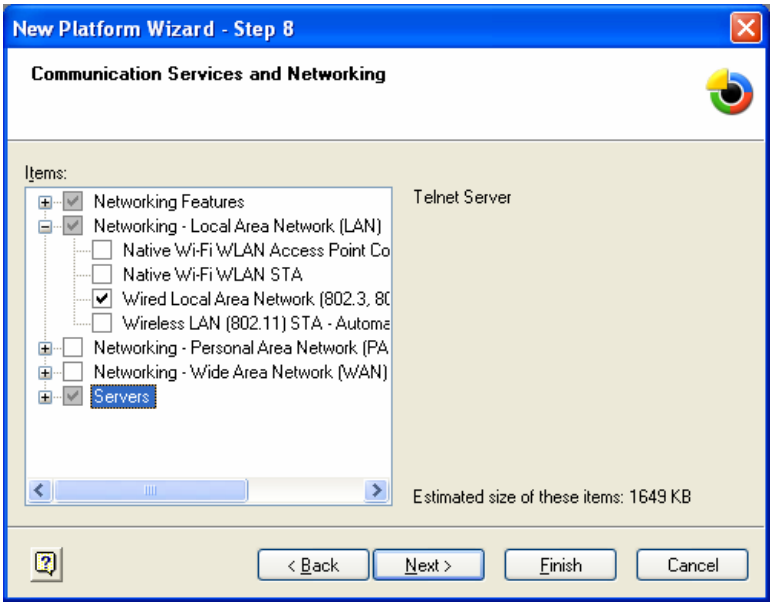


Figure 4-5: Communication services and Networking

Some registry information has been included for the FTP and the TELNET servers in the file *%_WINCEROOT%\Platform\CCXP\Files\Platform.reg*.

```
; TELNET SERVER
[HKEY_LOCAL_MACHINE\COMM\TELNETD]
    "IsEnabled"=dword:1
    "UseAuthentication"=dword:0      ;Don't use authentication

; FTP SERVER
[HKEY_LOCAL_MACHINE\COMM\FTPD]
    "IsEnabled"=dword:1
    "UseAuthentication"=dword:0   ;Don't use authentication
    "AllowAnonymous"=dword:1  ;Allow anonymous login
    "AllowAnonymousUpload"=dword:1    ;Allow anonymous upload of files
    "DefaultDir"="\\"  ;Root directory
```

These registry keys give you complete access to your target via TELNET and FTP, for demonstration purposes. In order to preserve the security of your targets, change these registry keys by enabling the *UseAuthentication* key and adding a list of allowed users.

For more information about security, read the chapters *Telnet Server Authentication and Security Considerations* of Windows CE on-line help.

Step 9: Device Management

Click *Next*.

Step 10: Storage Manager

In order to support any storage device used on your platform, you must include the support for the *FAT File System*.

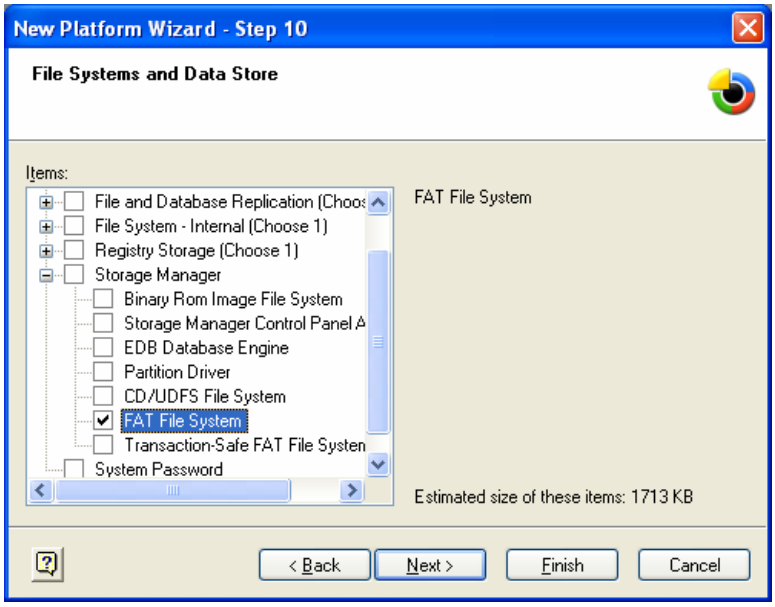


Figure 4-6: FAT File System and Data Store

Step 11: Fonts

Select support for fonts. Click *Next*.

Step 12: Language

Select support the different languages. Click *Next*.

Step 13: Browser Application

Select *Pocket Internet Explorer* then click *Next*.

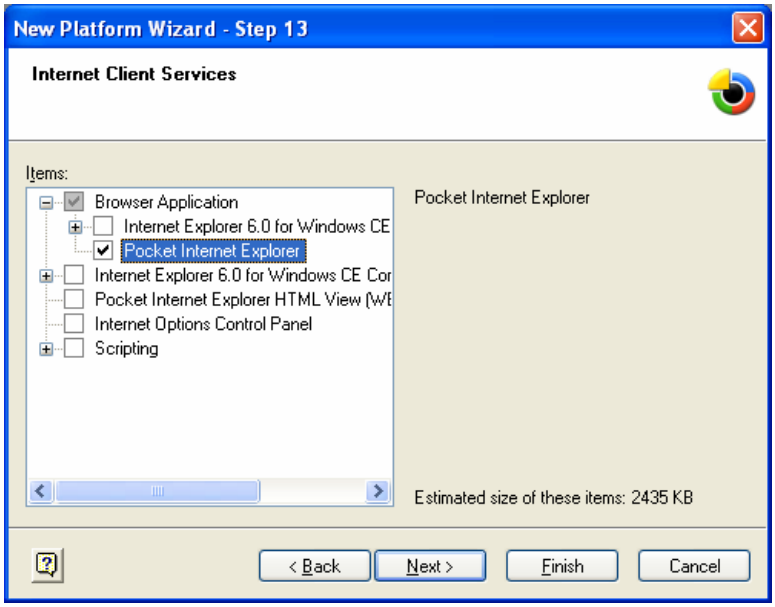


Figure 4-7: Internet Client Appication

Step 14: Graphics and Multimedia

You can select Graphics and Multimedia Technologies. Click *Next*.

Step 15: Security

Select Security. Click *Next*.

Step 16

Select *Shell* >Standard Shell > *Command shell.* From *User Interface,* select *Network User Interface, Quarter VGA Resources,* and from *Software Input Panel* select *Software-Base Input Panel, SIP for Small Screen* and *SIP for Large Screen.*

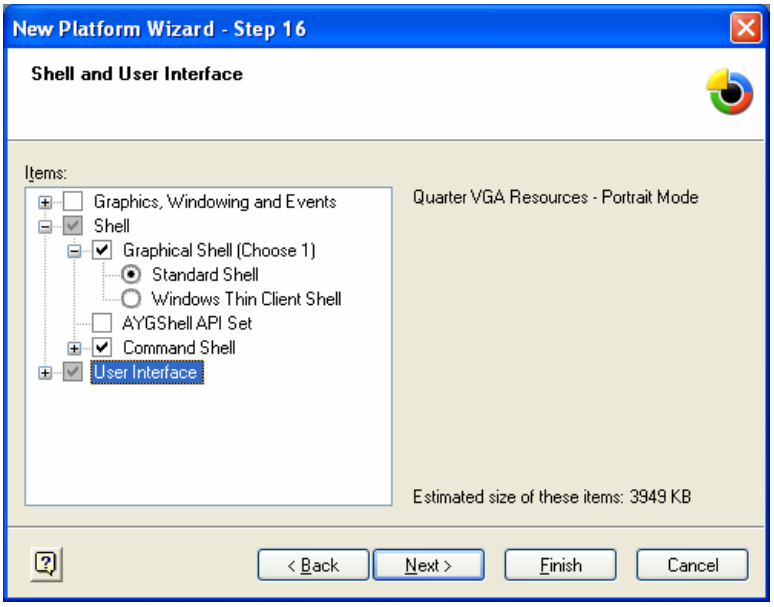


Figure 4-8: Shell and User Interface

Step 17: Error Reporting

Select Error Reporting. Click *Next*.

Step 18: Voice over IP Phone Services

Click *Next*.

After this step, help information may appear to warn you about security issues on your platform. Read this information carefully.

Figure 4-9: Security warnings

If you enable the checkbox “Notification acknowledged” you will not be warned again about this security issue. If you leave it unchecked, you will be reminded about this security warning.

After completion of the final step, click *Finish*.

Figure 4-10: Final step

You will see the features of your platform on the Platform window (left side of Platform Builder). Also, in the catalog window (right side) you can see the components of the CCXP module in the *Third Party* folder under *BSP*.

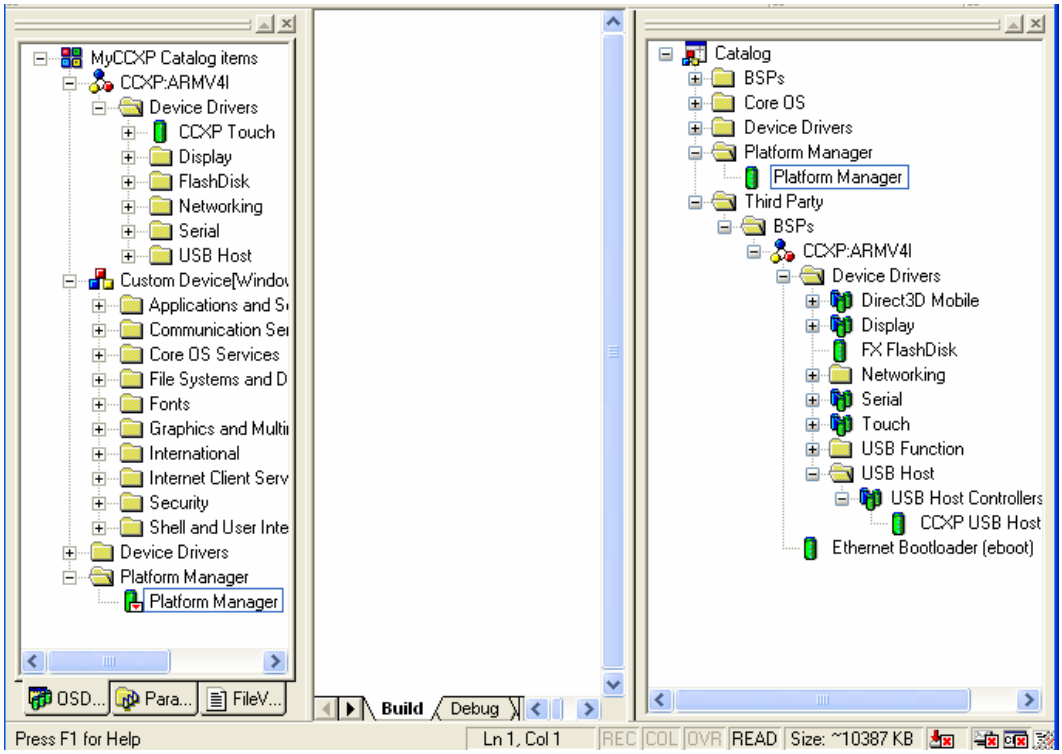


Figure 4-11: Platform view

As a final step, add the *Platform Manager* component to the project. The component is located in the catalog folder *Platform Manager* (right panel). You can click and drag it onto the platform window (left panel) or right-click and add it.

This component is necessary to perform application debugging with the embedded Visual C++ tool.

4.1.1. Platform settings

Before compiling the kernel, it is important to understand the different settings that can be changed in the platform. Go to *Platform* menu and click *Settings…*

4.1.1.1. Build options

For a *Release*[1] platform, only a few build options are enabled by default (*Target Control Support, Eboot space in memory, Full Kernel Mode* and *KITL*). For a *Debug* version, the *Kernel Debugger* is also enabled.

4.1.1.2. Language settings

Under the *Locale* tab, select the locales that your platform will support and the default language of your image.

The locales include information about currency formats, date and time formats, etc. specific to each country.

The default language specifies the language of the Windows CE user interface (buttons, menus, windows, etc.). For this example select *English US* or you will not see the links to some of the programs in your start menu.

[1] The *Release* version will create a smaller kernel but won't include information for debugging.

4.1.1.3. Environment Variables

The platform has several environment variables. Environment variables allow you to configure your platform and drivers. Some settings are made in the *CCXP.bat* file in the platform root directory. Others are set by each component and can be found at:

%_WINCEROOT%\public\common\oak\catalog\cec\CCXP.cec

The CCXP.bat file contains the following variables and default values:

```
set _TGTCPUVARIANT=XSC
set TARGET=270
set TFTPDIR=c:\TFTP-ROOT
set BSP_LAN91C111=1
set BSP_USBH_USE=1
set BSP_NOKEYPAD=1
set BSP_IP2REG=1
set BSP_BUILD_NK_NB0=1
```

_TFTCPUVARIANT and TARGET must not be changed. They specify the target platform to compile.

TFTPDIR must contain the path to the TFTP server's exposed folder configured in paragraph 2.2.1.

BSP_LAN91C111 is related to the Network driver SMSC91C111. If this variable is set to 1, the Network NDIS driver will be loaded. If it is not set, the Ethernet debugger will be used.

BSP_USBH_USE determines whether to use USB Host or Device functionality (both functions cannot coexist at the same time when using Starter Kit II base board). If it is set to 1, the USB Host functionality will be used. If it is not set, the USB Device functionality will be used instead.

You may remove USB support completely by setting the variable BSP_NOUSB=1.

BSP_NOKEYPAD is a variable that when set to 1 disables the keypad (this is the default configuration).

BSP_IP2REG is set to 1 to configure the NDIS driver with parameters established in the boot loader. To exclude the IP2REG NDIS registry configuration driver from the Kernel, set it to nothing. This will remove all registry entries for the driver.

BSP_BUILD_NK_NB0 is set to 1 to let Platform Builder create NB0 files that allow booting a Windows CE kernel directly with U-Boot without using EBOOT.

In addition, you can also use the following environment variables:

BSP_NODISPLAY can be set to 1 to exclude the display driver from the project.

BSP_NOSERIAL can be set to 1 to exclude the serial driver from the project. If the serial debugger is used, the variable must be set so that the serial port is disabled. Currently the system does not detect that automatically.

BSP_NOTOUCH can be set to 1 to exclude the touch screen driver from the Kernel. This will remove all registry entries and the BIB file entries for the driver.

4.1.2. Including virtual keyboard

Windows CE ships with a virtual software keyboard, which allows interaction with systems that do not have keyboard interface.

To include the keyboard into your platform, open the catalog on *Core OS > Shell and User Interface > User Interface > Software Input Panel*. Right click *Software-based Input Panel Driver* and then click *Add to Platform.*

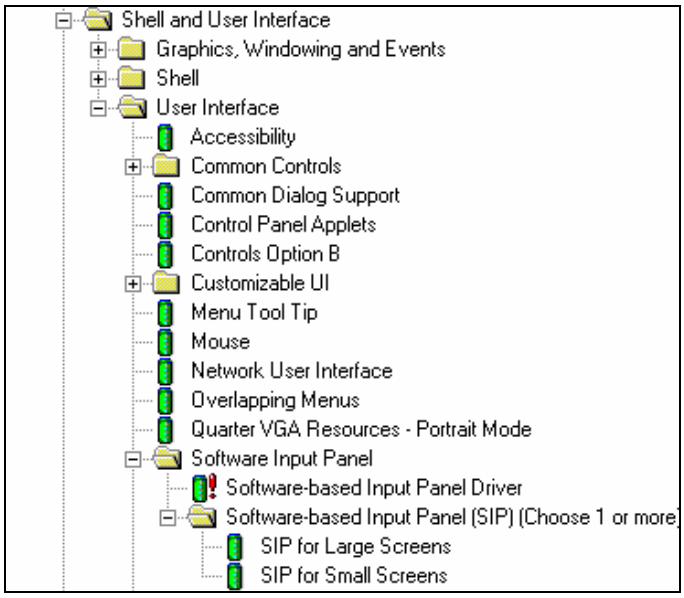


Figure 4-12: Adding virtual keyboard

On the catalog in the *Software-based Input Panel (SIP),* select one of the two software keyboards, either small or large, and add it to your platform. When you select the *Software-based Input Panel* one of the virtual keyboards is added automatically. Change it to the one you want in your project. The virtual keyboard will be accessible at the right side of your target's toolbar:

Figure 4-13: Virtual keyboard

The Input Method Selection Sample Application (*Sipselect*) code has been made into an integral part of the shell. This means that the *Sipselect* component no longer appears as a separate application. The *Sipselect* code is still available in the Samples folder.

%_WINCEROOT%\public\common\sdk\samples\sipselect

4.1.3. Including USB HID keyboard support

When USB Host support is selected for mouse and keyboard, a keyboard driver is needed in the project to generate the virtual key messages. We will use a standard Windows CE keyboard driver.

From the catalog view, move to *Device Driver\input Devices\Keyboard/Mouse* and select the *NOP (Stub) Keyboard* layout for the project.

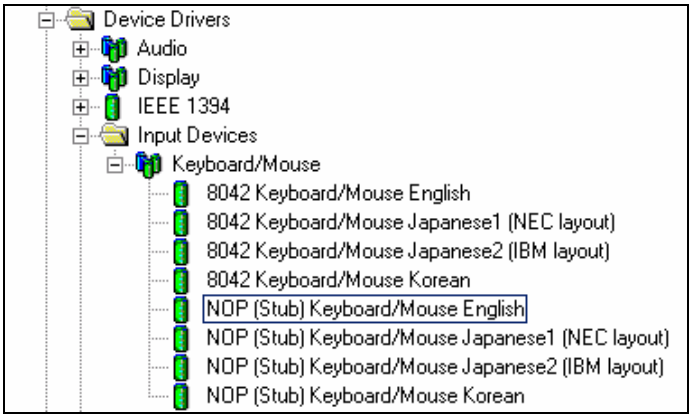


Figure 4-14: Adding Keyboard/mouse

The Stub keyboard driver has the basic functions to receive the scan code of a key and map it a character and virtual key code. By selecting *Add to OS Design,* the component will be included into the project.

4.2. Building the image

You can compile the *Release* or *Debug* version of your platform. In the first example, select a *Release* version.

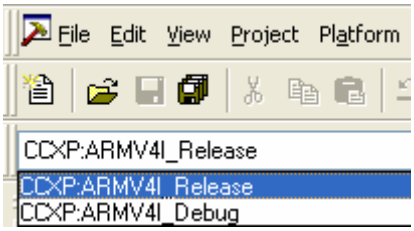


Figure 4-15: Version selection

Set in the *Build OS* menu the *Clean Before Building*, which cleans all the object files and always rebuilds all your drivers, and select *Sysgen*.

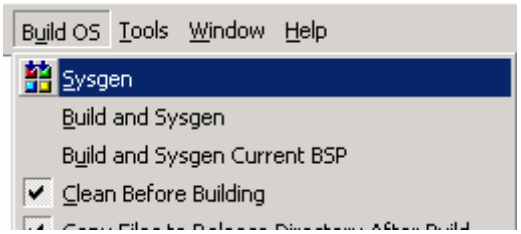


Figure 4-16: Sysgen

Sysgen is required the first time you build your project and every time you add or remove components.

The build process takes several minutes depending on the number of components you have included and the speed of your development PC.

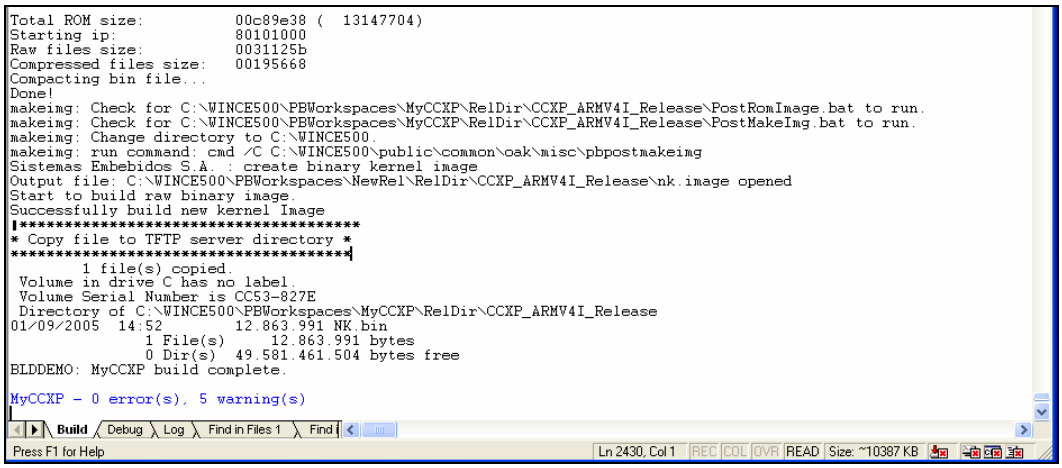


Figure 4-17: Build log

You may see some warnings due to Windows CE 5.0 fixing some DLLs that use the style from older versions.

After successful compilation[2], the Windows CE image (a file called NK.BIN) can be found:

%_WINCEROOT%\PBWorkSpaces\YourPlatformName\RelDir\CCXP_ARMV4I_Release

The size of the kernel image and the RAM size can be changed and configured in the *config.bib* file. This file is accessible from *%_WINCEROOT%\Platform\CCXP\Files.*

For changes made in the CCXP folder, you need to use the "Build and Sysgen Current BSP" menu item from the Build OS menu. It will recompile all sources in the platform folder and copy them to your project folder.

4.3. Downloading the image to the target

4.3.1. Boot process

4.3.1.1. Introduction

A boot loader is a small piece of software that executes soon after powering up a computer. On a desktop PC it resides on the master boot record (MBR) of the hard drive and is executed after the PC BIOS performs various system initializations. The boot loader then passes system information to the kernel and executes the kernel.

In an embedded system the role of the boot loader is more complicated because these systems do not have a BIOS to perform the initial system configuration. The low level initialization of the microprocessor, memory controllers, and other board specific hardware varies from board to board and CPU to CPU. These initializations must be performed before a kernel image can execute.

A boot loader for an embedded system provides the following features:

- Initialize the hardware, especially the memory controller.
- Provide boot parameters for the kernel.

[2] You may see some warnings that are due to Windows CE 5.0 fixing of some DLLs that use the style from older versions.

- Start the kernel.

Additionally, most boot loaders also provide convenient features that simplify development and update of the firmware:

- Reading and writing arbitrary memory locations.
- Uploading new binary images to the board's RAM via a serial line or Ethernet
- Copying binary images from RAM to Flash memory.

4.3.1.2. U-Boot

The universal boot loader, U-Boot, was adapted for the ConnectCore XP 270 module. The boot loader is capable of downloading Windows CE raw binary images or *NK.bin* via Ethernet to the target. Several default environment variables are available to make it more comfortable to launch, flash, or update kernel images.

When U-Boot is launched, it configures the serial console, the Ethernet interface, and the Flash memory and loads the settings stored as environment variables in the non-volatile memory.

Then it pauses a few seconds (programmable with *bootdelay* U-Boot variable) before it loads and starts the operating system image. You can stop the auto-boot process by sending a character to the serial port (pressing a key on the serial console connected to the target). If stopped, U-Boot displays a command line console similar to this:

```
U-Boot 1.1.3 (Nov  2 2005 - 12:07:39)  FS.2
for FS Forth-Systeme CCXP 270 on STK2

U-Boot code: A0080000 -> A00A2F74  BSS: -> A00E4324
RAM Configuration:
Bank #0: a0000000 64 MB
Bank #1: a4000000  0 kB
Bank #2: a8000000  0 kB
Bank #3: ac000000  0 kB
Flash: 32 MB
In:    serial
Out:   serial
Err:   serial

Detecting SMSC LAN91C111 Ethernet Controller...
Revision ID: 0x01
Chip ID: 0x09
SMSC LAN91C111 detected...
Get MAC Address from 1-wire EEPROM
MAC Address from EEPROM : 00:04:F3:00:38:10
Initializing ethernet...
Using MAC Address 00:04:F3:00:38:10
Hit any key to stop autoboot:  0
CCXP270>
```

To see the boot process, connect the provided serial cable between a serial port of your host PC and the serial port of the target. Then use a terminal program, like HyperTerminal, with the configuration 38400 / 8 / N / 1 and no Flow Control.

Features

- Download files using on board SMSC91C11 Ethernet controller
- Enable debugging over Ethernet or serial.
- Flash kernel images (NK.nb0).

- Configure the target system.

Additional considerations

The MMU is not set up by U-Boot; therefore, all memory addresses used by U-Boot are physical addresses.

The factory boot loader version installed in the target is in the *Bootloader* folder on the CD. To rebuild the boot loader, install the *Cygwin* tool chain and the boot loader sources from the BSP CD.

If the boot loader has been removed from Flash accidentally, the only way to get it back is using a debugger or separately available JTAG-Booster tool. For more information please refer to the debugger or JTAG-Booster documentation.

For more information about U-Boot commands and variables refer to chapter 12.1.

4.3.2. Windows CE boot methods

Windows CE is booted by U-Boot in one of the following ways:

- Platform Builder
- TFTP server
- USB storage device (e.g. a USB memory stick)
- Flash memory

Before you use one of the methods you need to verify that two boot loader variables are setup correctly. Set the following variables to met your network settings, were the first one is the IP addresses used for the target and the second from the machine you want to download the Windows CE kernel.

```
CCXP270> setenv ipaddr x.x.x.x
CCXP270> setenv serverip x.x.x.x
```

The following information describes each method.

4.3.2.1. Boot with Platform Builder

The boot loader comes with a script that will send BOOTME messages over Ethernet, connect to the Platform Builder, and download the kernel image (NK.bin).

Execute the following command on the U-Boot command shell to download your Windows CE kernel:

```
CCXP270> run connect_to_pb
```

On the Platform Builder side select *Connectivity Options* from the *Target* menu. This will open the Connectivity Dialog. If you have not specified a target device yet, click *Add Device* on the left side panel and enter a new name for the target device. The *Associated OS Design/SDK* can be set to none. Click Add. This will add your new name to the Target Device name list.

Click on *Kernel Service Map* and select your new added device from the Target Device list.

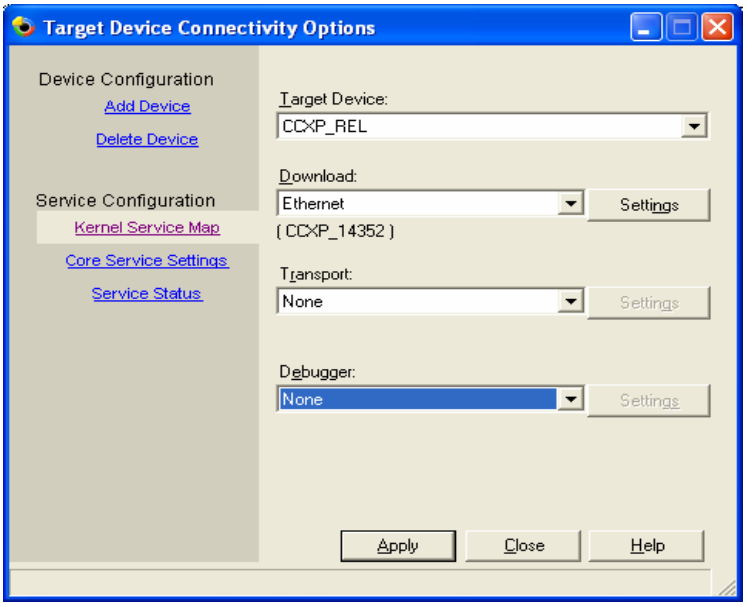


Figure 4-18: Target Device Connectivity Options

Then select Ethernet in the *Download* list. If you want to download the image (and only the image) the Transport and Debugger entries must remain None.

If you want to connect to the image after downloading, select Ethernet as Transport and KdStub as Debugger. Finally, click the *Settings* button beside the *Download* list. A new window will pop up:

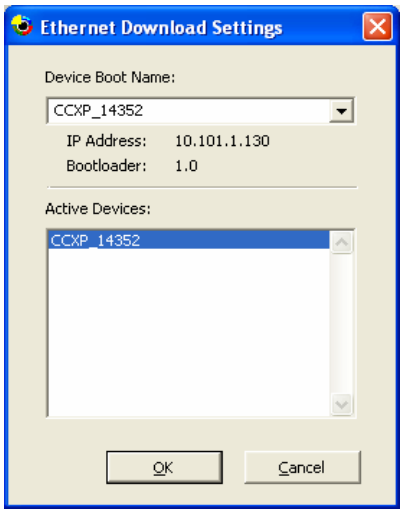


Figure 4-19: Ethernet Download Settings

The target has been sending BOOTME messages from the moment you executed the "run connect_to_pb" command from U-Boot. Platform Builder will catch these messages and display the name of your device. Select your device and click OK. The name of the current used device will be shown under the *Download* and *Transport* entries.

Apply the new settings, close the dialog, and select the new device from the Platform Builder *Active target* device list.

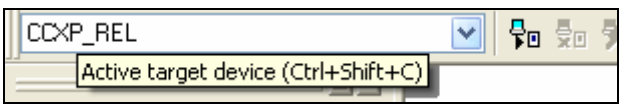


Figure 4-20: Active target device

Now you go to the *Target* menu and click *Attach Device*. This will download the kernel to your target device.

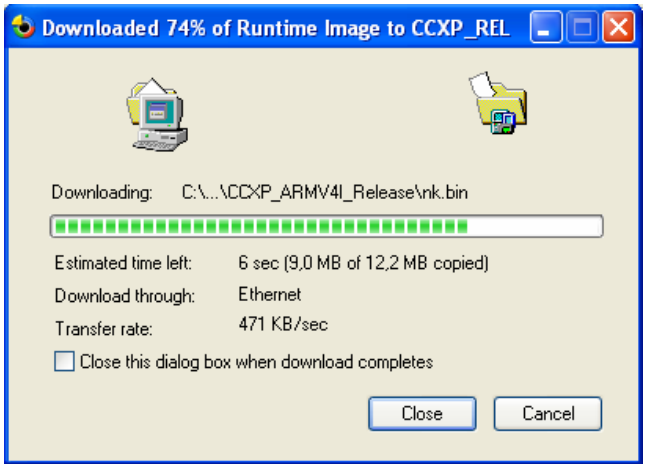


Figure 4-21: Downloading image to target

Type in the following commands in the U-Boot command shell and the target will automatically boot this way when the it powered up or reset:

```
CCXP270> setenv bootcmd run connect_to_pb
CCXP270> saveenv
```

4.3.2.2. Boot with TFTP Server

The boot loader comes with a script that manages the download of the image into RAM and jumps to the right memory location of the Kernel. This is achieved by U-Boot without the need or use of Platform Builder. The BSP generates both images, the NK.bin and a raw binary kernel image (called *wce-CCXP270*), with each build that can be downloaded via TFTP.

Start your TFTP server application. If you previously configured this folder in the *CCXP.bat* file, its exposed folder will already contain the Windows CE raw kernel image *wce-CCXP270.* Now you are ready to download the Windows CE raw binary kernel image over TFTP to the CCXP target.

Execute the following command from the U-Boot command shell:

```
CCXP 270> run boot_wce_net
```

To execute this function automatically each time the target is powered up or a reset is made, then type in the following commands:

```
CCXP 270> setenv bootcmd run boot_wce_net
CCXP 270> saveenv
```

4.3.2.3. Boot from USB memory stick

It is possible to load a kernel image from a USB storage device. To do so you need a FAT partition on the USB device. Then copy the Windows CE raw binary kernel image *wce-CCXP270* to it.

Once you have a Windows CE image in the USB memory stick, plug the stick into the USB host connector of your target and boot it with the following command:

```
CCXP270> run boot_wce_usb
```

To execute this function automatically each time the target is powered up or a reset is made, then type in the following commands:

```
CCXP270> setenv bootcmd run boot_wce_usb
CCXP270> saveenv
```

4.3.2.4. Boot from Flash memory

The boot loader is able to boot a Windows CE kernel that resides on the Flash memory. To enable this function you must store a Windows CE image in Flash.

Downloading and Writing the image to Flash

The boot loader comes with a script that downloads the Kernel image into RAM, formats the Flash, and copies the Kernel image from RAM into Flash. This is all achieved by U-boot without the need or use of Platform Builder. It uses the TFTP server to download the kernel image initially; therefore, you must be running your TFTP server.

Execute the following command on the terminal program to write your kernel to Flash:

```
CCXP270> run update_wce_tftp
```

Booting the image from Flash

Once you have a Windows CE image in the Flash memory, you can boot it with the following command:

```
CCXP270> run boot_wce_flash
```

To execute this function automatically each time the target is powered up or a reset is made, then type in the following commands:

```
CCXP270> setenv bootcmd run boot_wce_flash
CCXP270> saveenv
```

4.4. Updating Flash memory

This chapter describes how you can update the U-Boot boot loader and the Windows CE kernel image in the Flash memory of your module.

It is strongly recommended that you test your images before updating the Flash memory, by downloading them over Ethernet using Platform Builder, TFTP or USB as seen in chapters 4.3.2.1 4.3.2.2 and 4.3.2.3 respectively.

4.4.1. Updating a running system

On a running system, that is, a system that is able to start the boot loader, U-Boot contains predefined macros that can update the on-module Flash memory.

If the boot loader is corrupted, you will need a hardware debugger to restore it. See paragraph 4.4.2

Power up (or reset) the target. After 2-4 seconds, the boot loader messages appear on the serial port. Hit any key to interrupt the auto-boot process. From the U-Boot shell, you can update the Windows CE image in Flash and the boot loader.

4.4.1.1. Updating the boot loader

If you want to update the boot loader (with a new version or a modified version that you have created) First copy the new image file *u-boot-ccxp270stk2.bin* to the exposed folder of your TFTP server.

By executing the following command, the new boot loader image will be downloaded to the target via TFTP and then written to the Flash sector that holds the boot loader:

```
CCXP270> run update_uboot_tftp
```

4.4.1.2. Updating the Windows CE kernel

From TFTP server

You can download a new Windows CE kernel image via TFTP and write it to the Flash memory by executing the following command:

```
CCXP270> run update_wce_tftp
```

From a USB memory stick

You can download a new Windows CE kernel image from a USB memory stick and write it to the Flash memory by executing the following command:

```
CCXP270> run update_wce_usb
```

4.4.2. Updating a corrupted system

If the Flash memory has become corrupted and the system cannot boot anymore, then the Flash memory must be reprogrammed using the JTAG interface and the separately available JTAG-Booster.

This tool enables you to copy the boot loader much faster to the onboard Flash. Before storing a new boot loader to the CCXP module, the JTAG Booster hardware has to be connected to the target board.

Connect the JTAG Booster to the parallel port of your host PC and the other end to the JTAG connector on the target board (8-pin connector beside the DB25 connector). Open a DOS box on the host PC. Copy the boot loader binary image that you want to update and move it to the directory where the JTAG Booster software resides.

The JTAG-Booster software only recognizes 8+3 filenames lengths, therefore you must rename the default filename *u-boot-ccxp270stk2.bin* to, for example, *uboot.bin*

To update the boot loader image, type the following command:

```
C:\JTAG> jtag270 /p uboot.bin /driver=1 /lpt-base=378
```

Follow the instructions of the jtag270.exe program until the new boot loader is completely written to Flash. For more details please refer to the JTAG Booster documentation. When the boot loader is successfully downloaded, reset the target. Over the serial port, as described in the previous chapters, you should see the boot loader starting.

The thick wire on the JTAG-Booster connector is pin 2 and indicates ground. Pin 1 is connected to the square soldered pad on the development board.

5. BSP/Kernel Development with Platform Builder

5.1. Kernel Debugging

Ethernet debugging is implemented by not setting the environment variable BSP_LAN91C111 *CCXP.bat*, like this:

```
BSP_LAN91C111=
```

This must be made during compile time and cannot be changed during run time. At the same time this will enable the shared Ethernet functionality of Windows CE.

Now you can build a Debug Kernel. Follow the instructions described in paragraph 4.2 but select Debug instead of Release version:

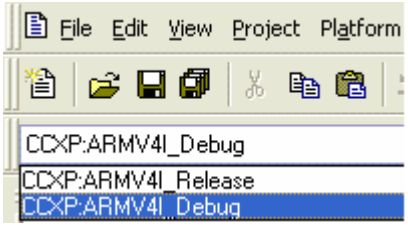


Figure 5-1: Version selection

When the kernel is built, you must configure the connection to the target. Follow the instructions described in paragraph 4.3.2.1 but this time select Ethernet and KdStub in the *Transport* and *Debugging* lists respectively:

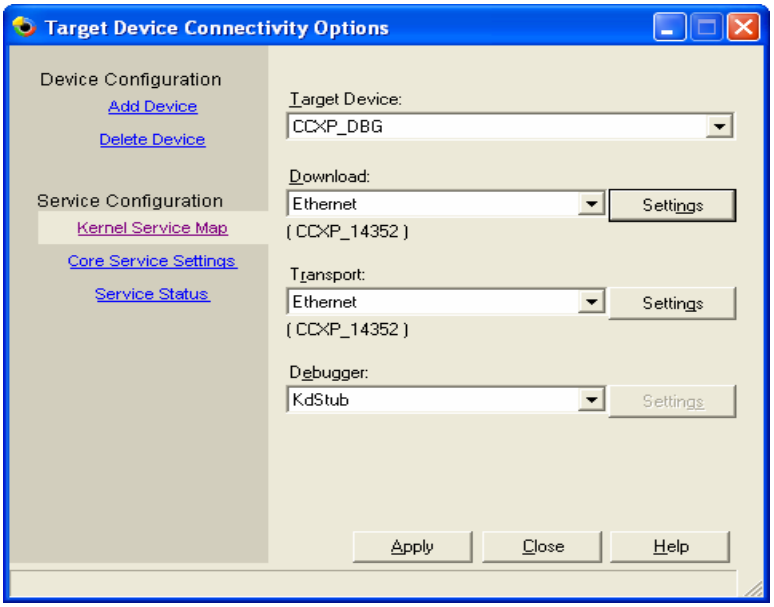


Figure 5-2: Target Device Connectivity Options

If you have previously connected Platform Builder to the target in the Release version, these fields may be disabled. To enable them you must first disconnect the target from Platform Builder. This is done from the menu *Target > Detach device*

Apply the new settings, close the dialog, and select the new device from the Platform Builder *Active target* device list.

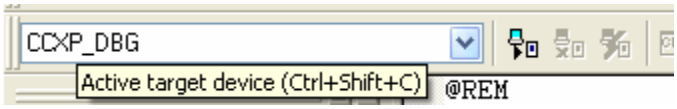


Figure 5-3: Active target device

Go to the *Target* menu on Platform Builder and click *Attach Device*.

Go to the target and execute the following command from the U-Boot shell:

```
CCXP270> run connect_to_pb
```

This will download the kernel to your target device.

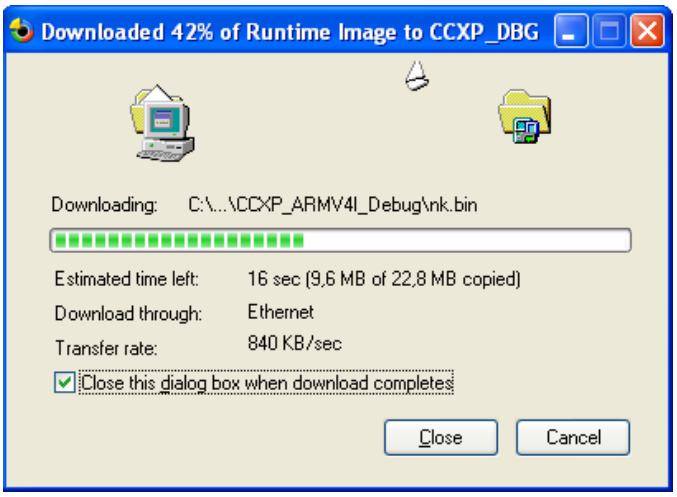


Figure 5-4: Downloading image to target

After downloading the kernel, the services CESH (Console Debug Shell tool) and CETerm (Target Messages) are initiated on the target. Windows CE starts without additional tools.

```
The Kernel Debugger is waiting to connect with target.
UP2OCR value = 0x20000
UP2OCR value = 0x3000c
Setup USB Port2 for USB Host.
-OEMInit
Error Reporting Memory Reserved, dump size = 00020000
Booting Windows CE version 5.00 for (ARM)
&pTOC = 80109d2c, pTOC = 818423a4, pTOC->ulRamFree = 818d2000, MemForPT = 00000000
Configuring: Primary pages: 9994, Secondary pages: 0, Filesystem pages = 4997

Booting kernel with clean memory configuration:
Memory Sections:
[0] : start: 818d3000, extension: 00003000, length: 0270a000
Sp=ffffc7cc
Windows CE KernelInit
Updated eptr->e32_vsize to = 000bd000
Scheduling the first thread.
0x83fdf024: KernelInit2: pCurThread=83fdf024 hCurThread=03fdf266 hCurProc=03fdf002, KernelInit = 80128cbc
0x83fdf024: Detecting VFP...
0x83fdf024:  VFP Not Found!
0x83fdf024: Updated eptr->e32_vsize to = 00006000
0x83fdf024: Updated eptr->e32_vsize to = 00029000
0x83fdf024: +OEMIoControl(0x1010004, 0xc201fc2c, 4, 0x818ccdb0, 520, 0x818ce7c0)
0x83fdf024: +OALIoCtlHalGetDeviceInfo(...)
0x83fdf024: -OALIoCtlHalGetDeviceInfo(rc = 1)
0x83fdf024: -OEMIoControl(rc = 1)
0x83fdf024: +OEMIoControl(0x1010004, 0xc201fc2c, 4, 0x818ccba8, 520, 0x818ce7bc)
0x83fdf024: +OALIoCtlHalGetDeviceInfo(...)
0x83fdf024: -OALIoCtlHalGetDeviceInfo(rc = 1)
0x83fdf024: -OEMIoControl(rc = 1)
0x83fdf024: +OEMIoControl(0x1010004, 0xc201fc2c, 4, 0x818cc9a8, 32, 0x818ce7b8)
0x83fdf024: +OALIoCtlHalGetDeviceInfo(...)
0x83fdf024: -OALIoCtlHalGetDeviceInfo(rc = 1)
0x83fdf024: -OEMIoControl(rc = 1)
0x83fdf024: Updated eptr->e32_vsize to = 00009000
0x83fdf024: Updated eptr->e32_vsize to = 00022000
0x83fdf024: Starting kernel debugger software probe (KdStub) - KD API version 18
0x83fdf024: >>> Loading module NK.EXE at address 0x80100000-0x801BD000 (RW data at 0x81856000-0x818B713F)
Kernel debugger connected.
The Kernel Debugger connection has been established (Target CPU is ARM).
Target name: CCXP
```

Figure 5-5: Kernel debug messages

Now stop the target and debug the kernel. The debugging windows and menu items in the Platform Builder IDE allow looking at processes, threads, and other target debugging information like watch variables, dump the memory, etc.

The target can be stopped with the menu item *Debug* > *Break*.

When the target is stopped, a source file is opened where the system halted.

With <F10> you can step through the code and see the behavior of a driver or an application. If you want to run through the rest of the code press <F5> (Go).

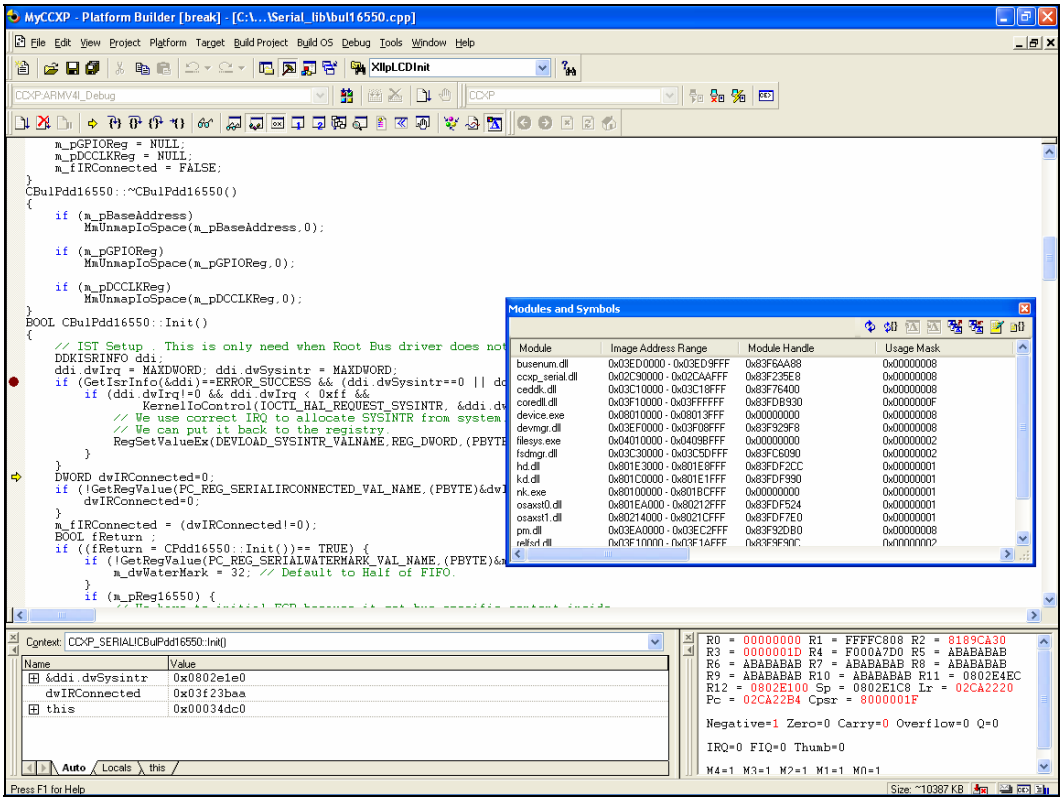


Figure 5-6: Kernel debugging

During loading of the debug kernel, the following dialog will pop up. Mark the check box at the bottom

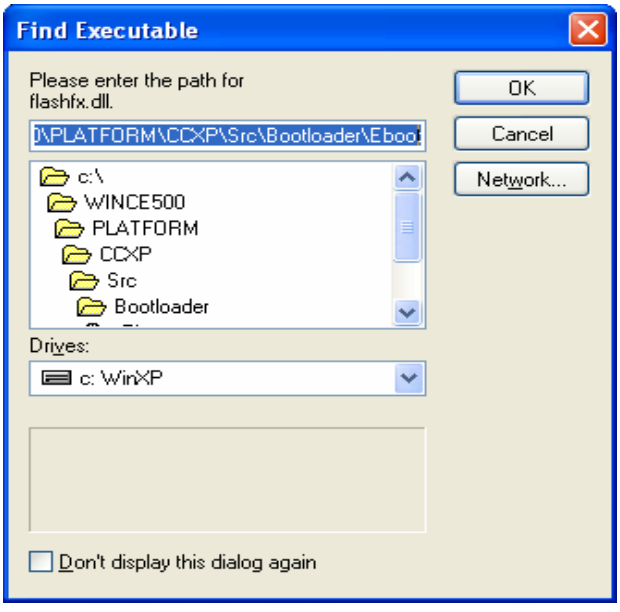


Figure 5-7: Find Executable

of the dialog and then select *Cancel*.

5.1.1.1. Sharing Ethernet Debugging Services

Debugging the kernel and applications usually takes place over the Ethernet interface. However, there is the possibility to share the Ethernet interface between the debugging network and application network traffic (a socket application, telnet sessions, etc.). The main registry entries needed for this can be found in the *COMMON.REG* file.

The debugger will use the IP address that the target uses to download the Kernel image.

The network interface for applications will use the IP address saved in U-Boot if the Windows CE environment variable IP2REG=1. If IP2REG is not enabled, the network interface for applications takes the IP from the following entries of PLATFORM.REG:

```
IF BSP_NOSHAREETH !
[HKEY_LOCAL_MACHINE\Comm\VMINI1\Parms\TcpIp]
    "EnableDHCP"=dword:0
    "DefaultGateway"="0.0.0.0"
    "UseZeroBroadcast"=dword:0
    "IpAddress"="0.0.0.0"
    "Subnetmask"="0.0.0.0"
    "DNS"="0.0.0.0"
ENDIF BSP_NOSHAREETH !
```

To include the shared Ethernet support you have to set the environment variable BSP_NOSHAREETH to nothing (default value).

5.1.1.2. Run Time Debugging

Under certain circumstances, because of some time critical aspect of the system, you may not want to debug step by step. Most of the drivers in Windows CE have a variety of Debug messages. These messages are divided into different zones. These zones (a maximum of 16) can be activated or deactivated. One method to set up the debug zones is to alter the registry of your host PC where you are designing the Kernel.

First open the Registry of your development PC by running *RegEdit.exe*. To start the program open *Run* from the *Start* menu. Enter *RegEdit* and select *OK*.

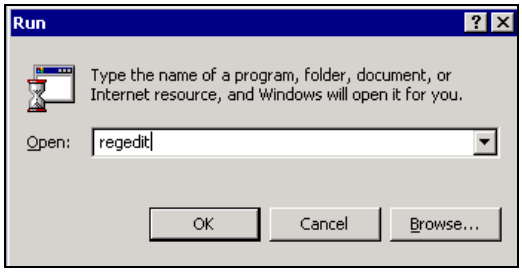


Figure 5-8: Edit the PC's registry

The registry editor will be opened. Go to folder *HKEY_CURRENT_USER\Pegasus\Zones*. Here you can create a new entry for the driver or component you want to specifically debug.

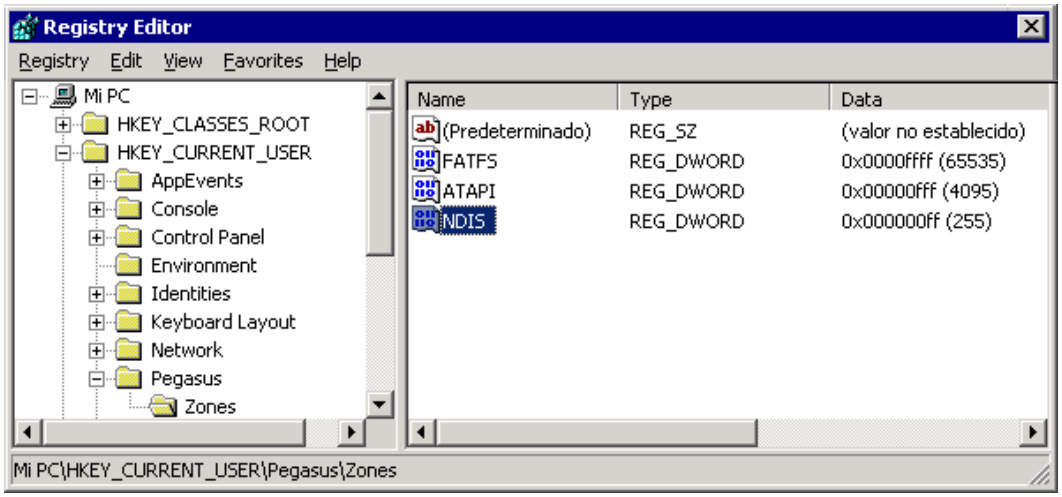


Figure 5-9: Registry Editor (Windows CE debug zones)

The names that you should enter here can be found in the *DBGPARAMS* statements of the corresponding driver or component sources. For more information, please refer to the Online Help. Most of the sources of Windows CE are located at:
%_WINCEROOT%\Public\Common\Oak\Drivers

Here you should find the proper names for most of the drivers supported by Windows CE. The values that you set for each entry depend on the amount of zones that you would like to activate. E.g. the entry FATFS, in the picture above, activates all 16 zones. Each bit represents a debug zone (activated with a 1 and deactivated with a 0).

5.2. Remote Tools

The Remote Tools allow you to monitor your target right from your desktop development environment. You can edit the target registry, view the processes that are running, or take a snapshot of the target's screen, etc.

If you want to use the Remote Tools in a Release version, you have to add the *Platform Manager* to your project. The component can be found in the Catalog view of the Platform Builder. Simply move in the Catalog view to the component, right-click it and select *Add to Platform*.

Figure 5-10: Adding Platform Manager

This will include the three possible media to communicate with the target:

- ActiveSync
- KITL
- Manual server

In a Debug version you don't need to include the Platform Manager to be able to use the Remote Tools, since they take place by means of the KITL connection established for debugging.

If you are working with a Release version and Manual server, a dialog will come up and ask to verify that certain files are on the target side:

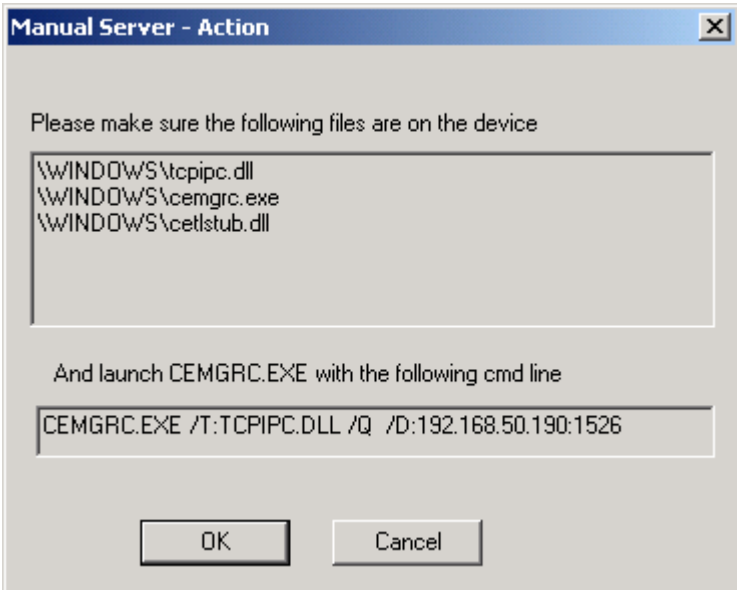


Figure 5-11: Manual Server action

Select and copy the *CEMGRC.EXE* line but **do not click the OK button yet**. Open a telnet session to the target and paste the command line to the telnet session as shown in this picture:

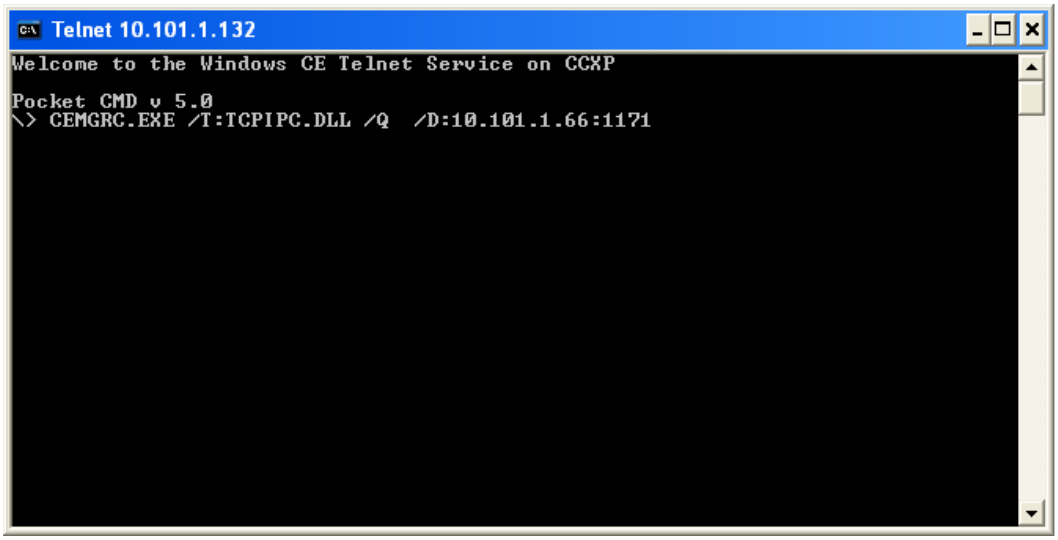


Figure 5-12: Paste CEMGRC.EXE line to telnet session

Press ENTER so the target starts listening for connections from the Remote Tools.

Now you can click the *OK* button in the dialog shown in Figure 5-11. The target and Remote Tools will start to communicate.

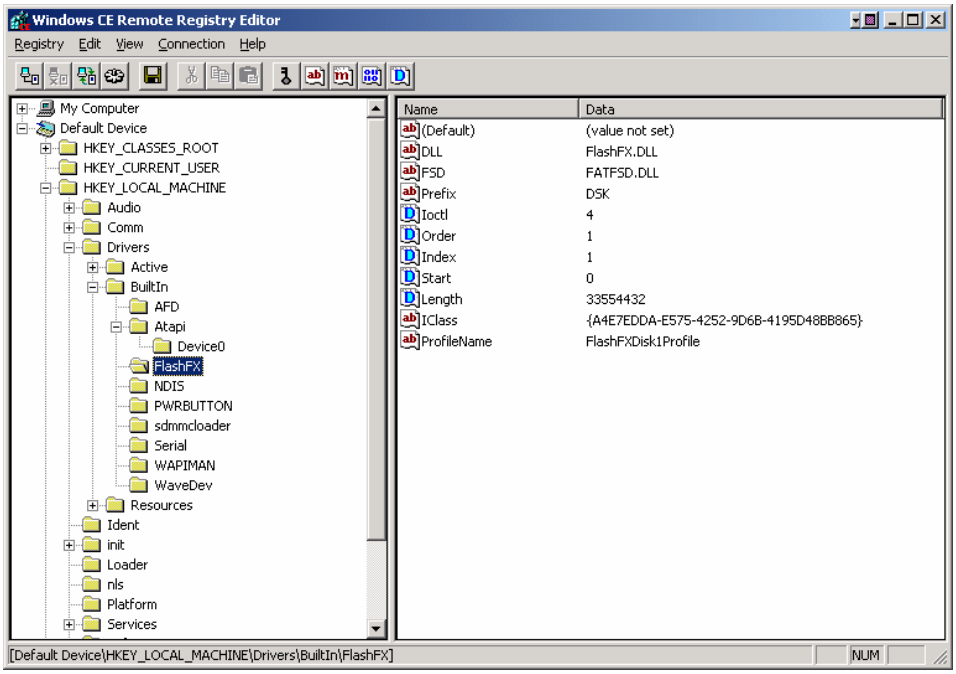


Figure 5-13: Remote Registry Editor

6. Application Development with Embedded Visual C++

The following chapter will guide you through the process of creating applications with Embedded Visual C++. You need to have installed all service packs for the Embedded Visual C++ to work correctly under Windows CE 5.0.

6.1. Creating a new project

In this example, you will create a console application for the Kernel that we have previously made and with the SDK for that particular Kernel. Before starting ensure the SDK is installed on the host PC. Then open the embedded Visual C++ and select from the *File* menu the item *New*.

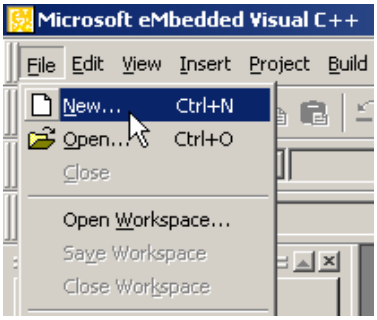


Figure 6-1: Create new project

Select *WCE Application* and enter the name. At the bottom right corner of the dialog select the *ARMV4I* CPU and click *OK*.

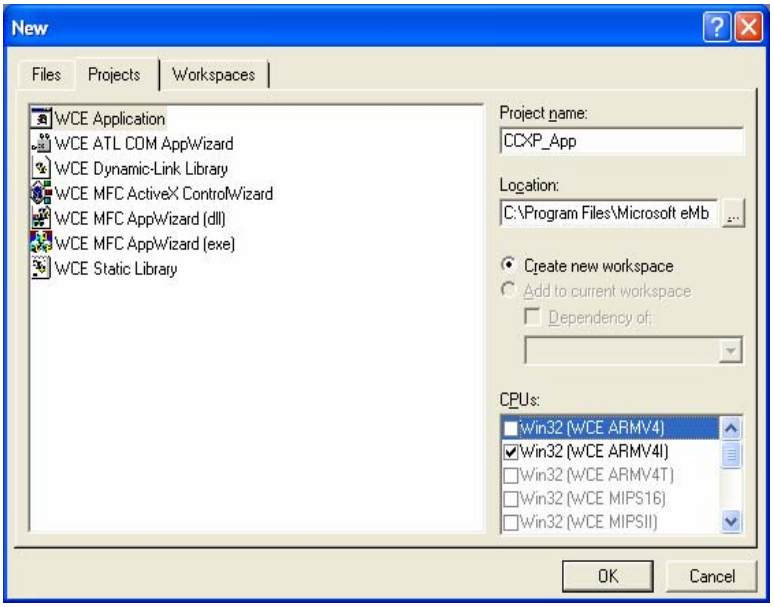


Figure 6-2: New project dialog window

On the next screen select *A typical Hello World Application* and click *Finish*.

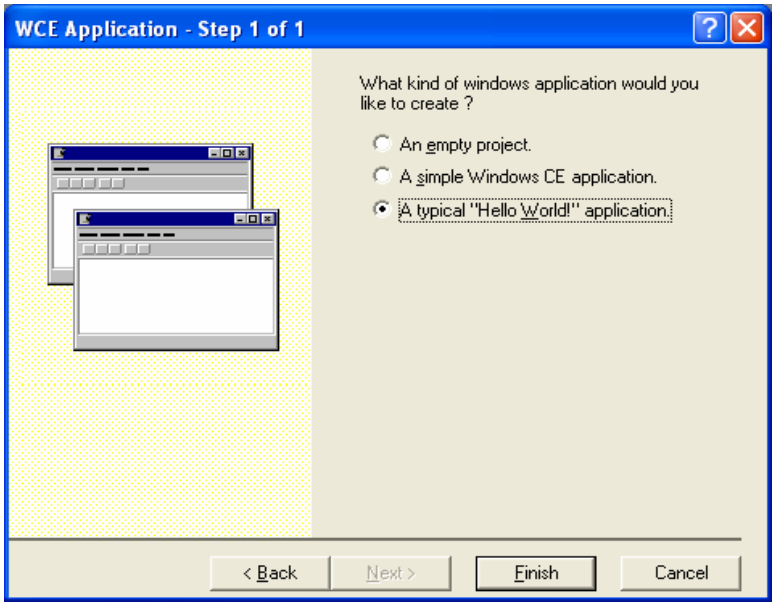
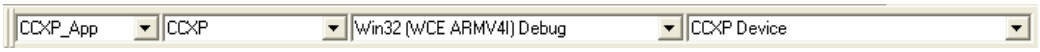


Figure 6-3: Application template

The wizard will create all files and sources for a small graphical application.

When you build the application, you will see three warnings that you should ignore. Immediately after building, embedded Visual C++ will try to connect to the target device. It will use the configuration of the device that is selected in the *Default Device* List.



By default the transport is set to *manual server*. If you want to change the default, select *Configure Platform Manager* under the *Tools* menu.

When you have downloaded the application, execute it over the embedded Visual C++ or launch it directly from the target.

Figure 6-4: Hello world application running on the target

The application will be placed in the *Windows* folder of the target device.

6.2. Downloading the application to the target

When the build has successfully completed, embedded Visual C++ will automatically try to download the application to the target device. Because you have selected a specific SDK, the embedded Visual C++ has selected the corresponding device configuration. A dialog will come up and ask you to verify certain files are on the target side.

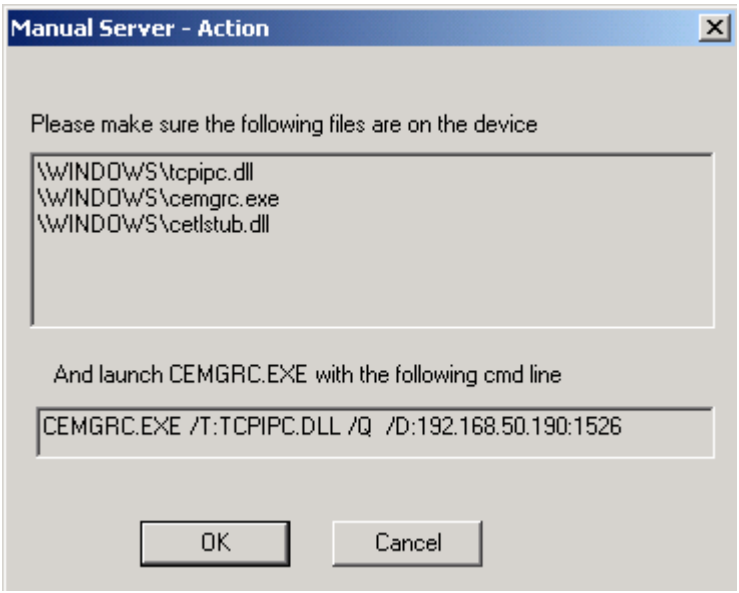


Figure 6-5: Manual Server action

Select and copy the *CEMGRC.EXE* line but **do not click the OK button yet**. Open a Telnet session to the target and paste the command line to the Telnet session as shown in this picture:

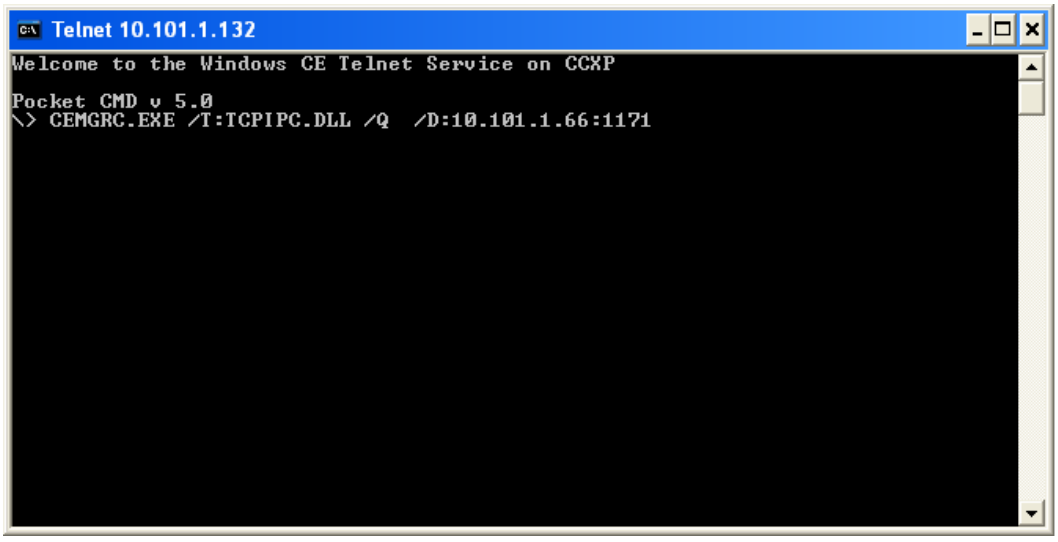


Figure 6-6: Paste CEMGRC.EXE line to telnet session

Press ENTER so that the target starts listening for connections from embedded Visual C++.

Now you can click *OK* in the dialog shown in Figure 6-5. The target and embedded Visual C++ will start to communicate and download the application and execute it. The download is made once to

the target. You can control the application directly by your target or even execute it by embedded Visual C++. If you have made a change in the application, you need to repeat the steps listed above to download the application to the target.

6.3. Debugging the application

Open a source file of your application and set a breakpoint in a code line. Download the application as seen in paragraph 6.2. From the *Build* menu select *Start Debug > Go*.

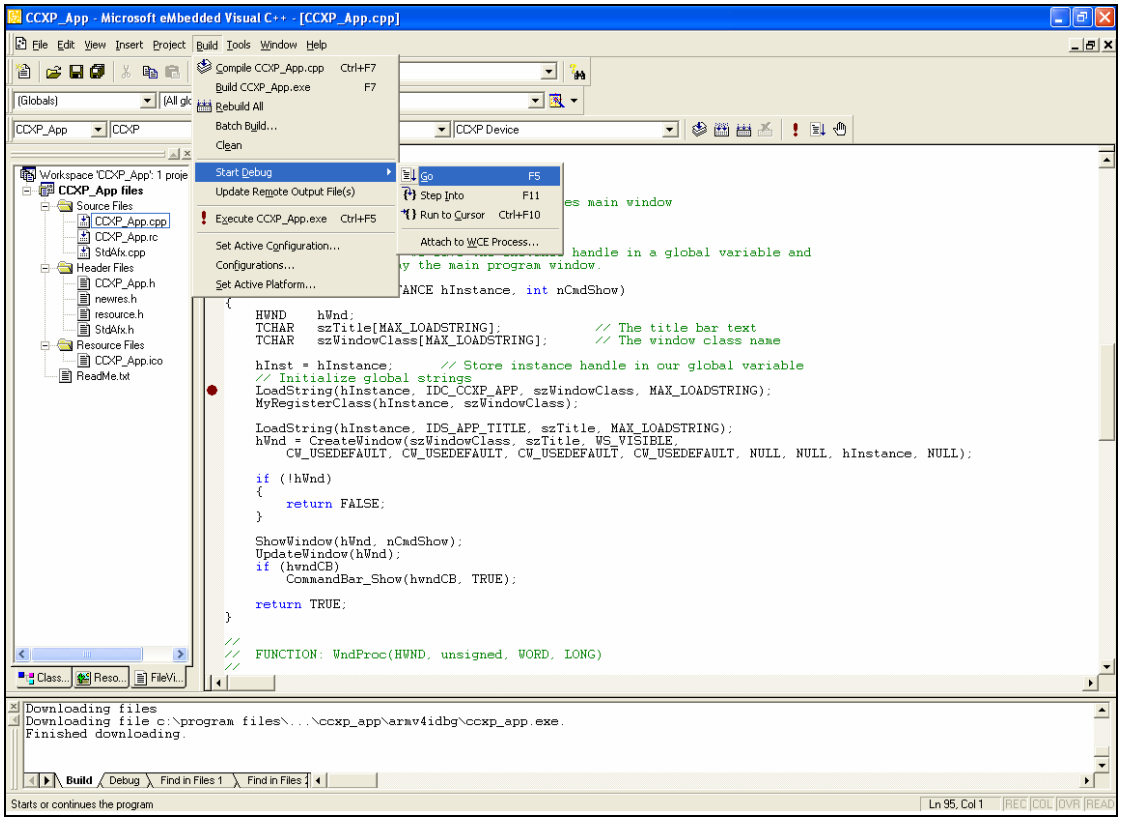


Figure 6-7: Launch the application for debugging

When the download completes embedded Visual C++ will open the debugger interface and will stop at the breakpoint you have set.

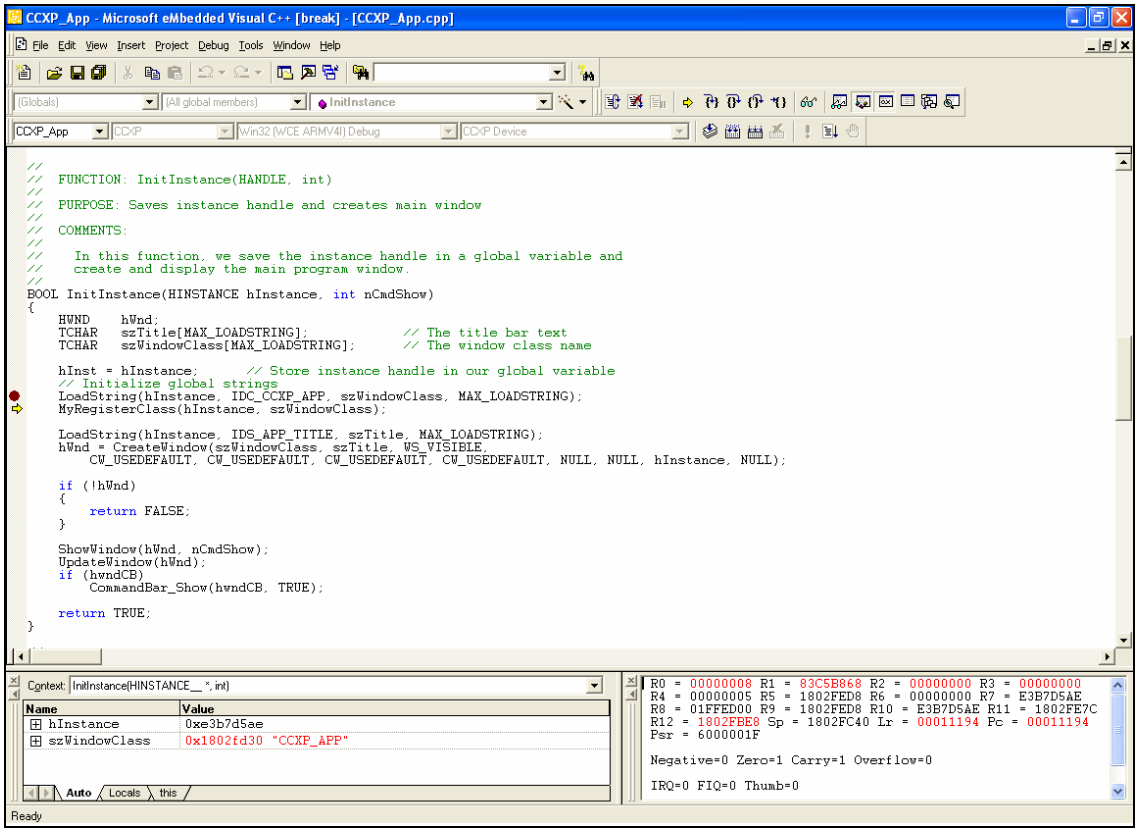


Figure 6-8: Application debugging

You can do single stepping or watch variables or the processor register. When you terminate the application, the debugging interface is closed automatically.

6.4. Modifying the Platform Manager Configuration

When you want to use a different transport for your device, you need to change the *Platform Manager* configuration. Under the *Tools* menu select *Configure Platform Manager* item.

A dialog will open with all the devices that can be configured and used. Select the device whose configuration you want to change and click *Properties*.

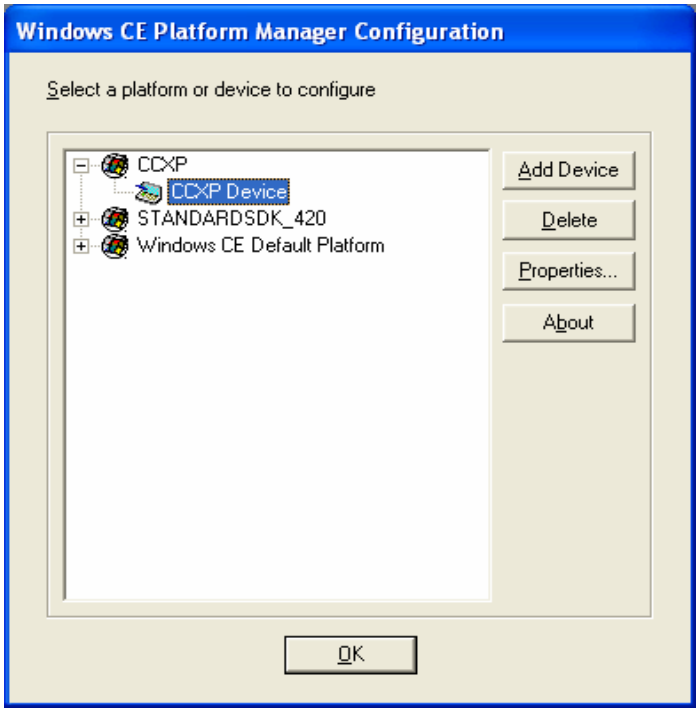


Figure 6-9: Platform selection

The Platform Manager supports three different transport and startup servers. Select the corresponding transport and startup server you want to use for your target device.

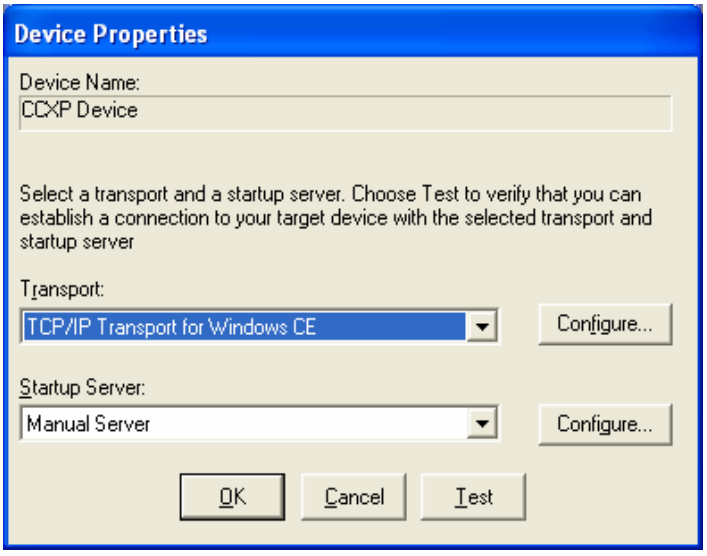


Figure 6-10: Transport and Startup server

Click *OK* to confirm the changes and use the new settings.

7. Advanced Topics

7.1. Creating a Software Development Kit (SDK)

The CD already contains an SDK made with the supported components of the example project. If your application needs more functionality or different components, you should make a new SDK related to all the components inside your Kernel.

This section describes the necessary steps to create the new SDK representing your specific project. With this SDK, you will be able to create applications for your Microsoft Windows CE platform.

Under *Platform* > *SDK* > *New* SDK, the *SDK Wizard* will guide you through the process of configuring and building a software development kit (SDK) for Microsoft Visual Studio .NET, Smart Device Extensions for Visual Studio .NET, and Microsoft eMbedded Visual C++ 4.0. The process generates a Microsoft Windows Installer (.msi) file that contains the necessary header files, libraries, Platform Manager components, run-time files, platform extensions, and documentation to generate a new application for you specific kernel.

Enter the properties for your SDK to uniquely identify it.

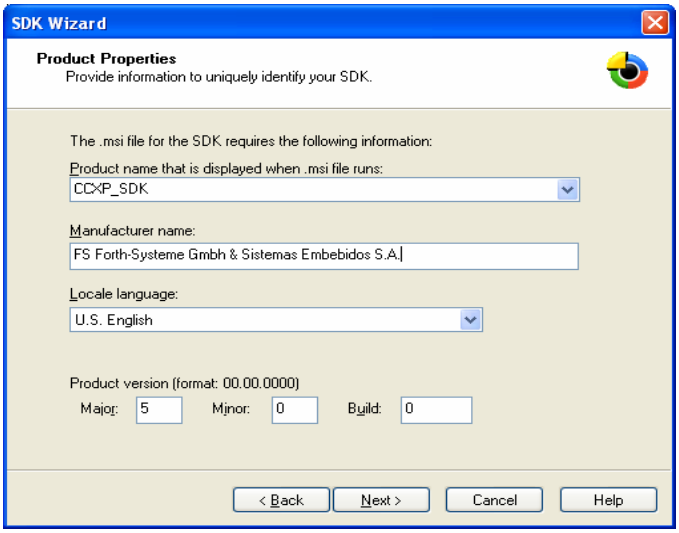


Figure 7-1: SDK properties

Select the development languages features you want to support.

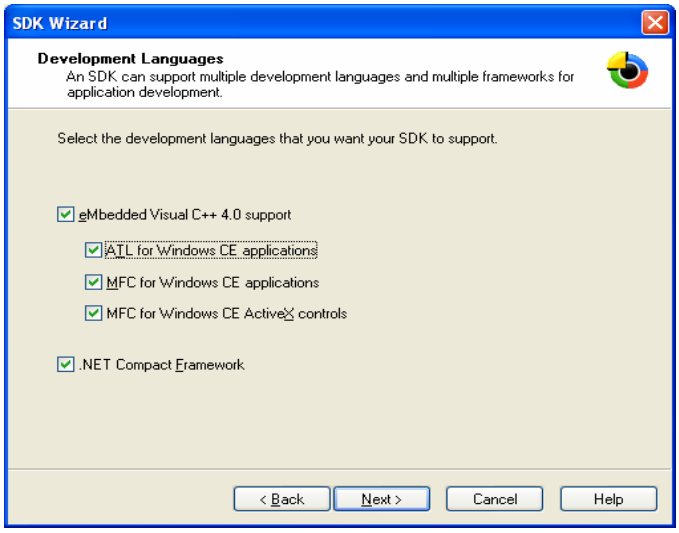


Figure 7-2: SDK languages support

Click Next and then Finish.. The configuration of the new SDK is located at *%_WINCEROO%\PBWorkspaces\YouProject\MakeSDK*

Some of the BSP drivers automatically export header files to the SDK in case you want to have access to them from your applications. If you need some additional headers, you can configure the SDK accordingly:

Go to menu *Platform > SDK > Configure SDK* and click on the *Additional Files* tab.

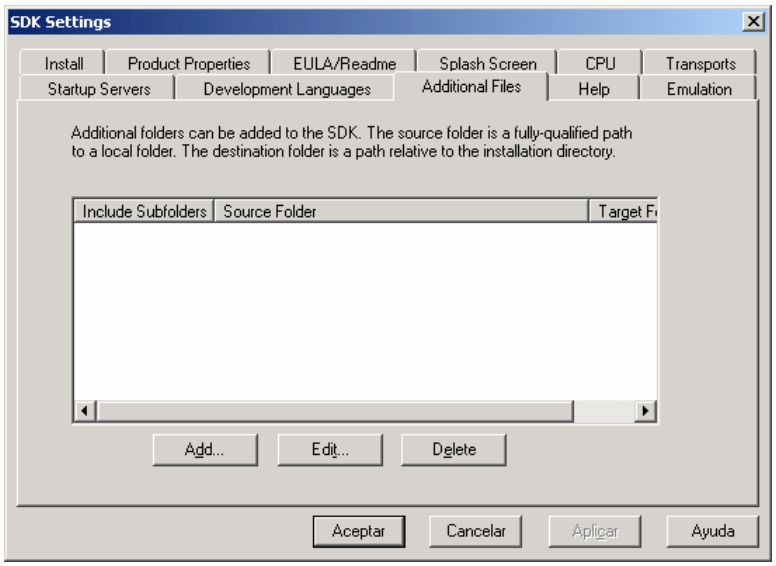


Figure 7-3: Include additional files in the SDK

Then add a new entry.

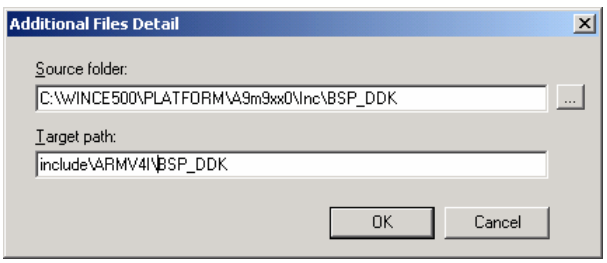


Figure 7-4: Include a file to the SDK

Once you've configured and added additional files to the SDK let's proceed to build it. To build the SDK go again to *Platform > SDK* and then select *Build SDK*.

This will generate a self-installable MSI file on *%_WINCEROOT%\PBWorkspaces\YourProject\SDK\YourProject_SDK.msi*

Double click this file to install the SDK into your host PC. You can also include further libraries or components to your SDK.

There is an already built SDK for the previous sample platform on the provided CD, within the folder \SDK. Once you have it installed you can begin to develop your embedded visual applications for your platform. If you are changing the Kernel image by including or removing components, you should make a new SDK for a secure application development.

Be sure that your embedded Visual Studio installation contains all Software Packages available by Microsoft. If it is not complete, the installation of the SDK may fail. For Visual Studio .NET 2003 you need to install the *Windows CE add-in*.

7.2. Insert existing projects

If you made an application with another project you can enter this project into your current project. To do so, move to the *Project* menu item of the Platform Builder, *Insert > Existing Project*.

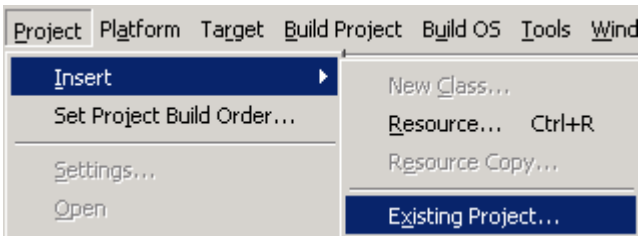


Figure 7-5: Insert Existing Project

A dialog box will open. Select the *.pbpxml file from the directory you want to include your sources from. Confirm with *Open* and the project will be included in your platform. The output (*.exe) will be located in the sub directory where you copied the sources, either Release or Debug.

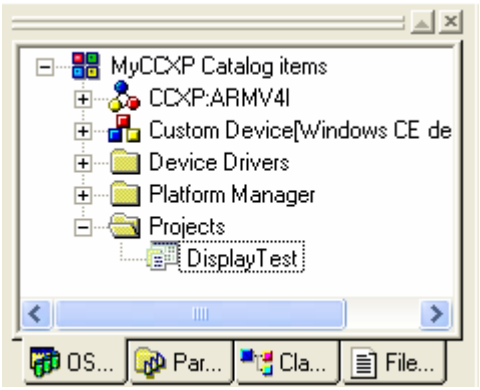


Figure 7-6: Inserted project

To edit the sources double click on the component of the *OSDesignView* or click on the *FileView* tab and find the file there.

7.3. Driver test applications

A set of applications have been developed to simplify the testing of driver functionality.

The source code is available to use as a reference for your application development.

Most of the drivers tested here will be 'Stream Interface Drivers' and are used by Open, Read, Write, IOCTL… other can have a different API (like audio or LCD drivers)

Those that are not WinCE standard (like GPIO driver) define the API in header files at *%_TARGETPLATROOT%\src\inc\BSP_DDK* folder.

Most of the driver test applications require that a driver is already loaded. Be sure that all environment variables related with that driver are correctly set. When the target is running you can check which drivers have been loaded using the registry editor and can look into key: *[HKEY_LOCAL_MACHINE\Drivers\Active]*

7.3.1. GPIO test

To include the GPIO test, insert the test project (*.pbpxml) from the *Apps\testGPIO* folder on the CD as explained in chapter 7.2. The new generated image will contain the application. To run it, you have several possibilities:

- If you have an Ethernet NDIS driver (or VMINI) and Telnet server available:
  - Open a Telnet session to your target with 'telnet xxx.xxx.xxx.xxx'
  - Execute the test with the following command:

```
\> testGPIO config output 16 0x00
```

- If you have LCD and touch screen and have included the 'Software Input panel' component:
  - Open a command window.
  - Execute the test with the command: **testGPIO config output 16 0x00**
- If you have connection with the target from Platform Builder:
  - Open 'CE Target Control' window.
  - Execute the test with the 's' command: **s testGPIO config output 16 0x00**

7.3.1.1. Test GPIO driver functionality

Here are the possible arguments that the testGPIO application admits:

Arg1	Arg2	Arg3	Arg4	Explanation
config	input	<PinNumber>		Configures PinNumber as input
config	output	<PinNumber>	[DefaultVal]	Configures PinNumber as output. If no Default value is specified, it is set to 0.
config	irq	<PinNumber>	[flags]	Configures PinNumber as an irq. Flags stablish the detection method. Bit0: FLAG_INTR_LEVEL 0 = Edge detect

				1 = Level (not valid in CCXP) Bit1: FLAG_INTR_INVER 0 = High level/edge 1 = Low level/edge
read	<PinNumber>			Reads current value of a pin configured as input, output or irq
set	<PinNumber>	<Value>		Sets the value of a pin configured as output
wait	<PinNumber>			waits until an interrupt happens on that pin (pin must be configured as irq)

Table 7-7: Arguments of testGPIO application

And the following table shows some examples:

Command	Description
testGPIO config input 17	Configures GPIO 17 as input
testGPIO config output 16	Configures GPIO 16 as output (no default value, =0)
testGPIO config output 16 1	Configures GPIO 16 as output with default value =1
testGPIO config irq 17 0	Configures GPIO 17 as IRQ on rising edge
testGPIO config irq 17 2	Configures GPIO 17 as IRQ on falling edge
testGPIO read 17	Reads GPIO 17 value
testGPIO set 16 1	Sets GPIO output 16 to a value of 1
testGPIO wait 17	Waits for an interrupt on IRQ configured GPIO 17.

Table 7-8: Examples of testGPIO application

7.3.1.2. Suggested test

As an example, we'll do a test with GPIO 16 and 17 on the Starter Kit II.

Start connecting GPIO16 to GPIO17 by means of a jumper (they are pins 16 and 17 of the X12 connector, according to the base board documentation):

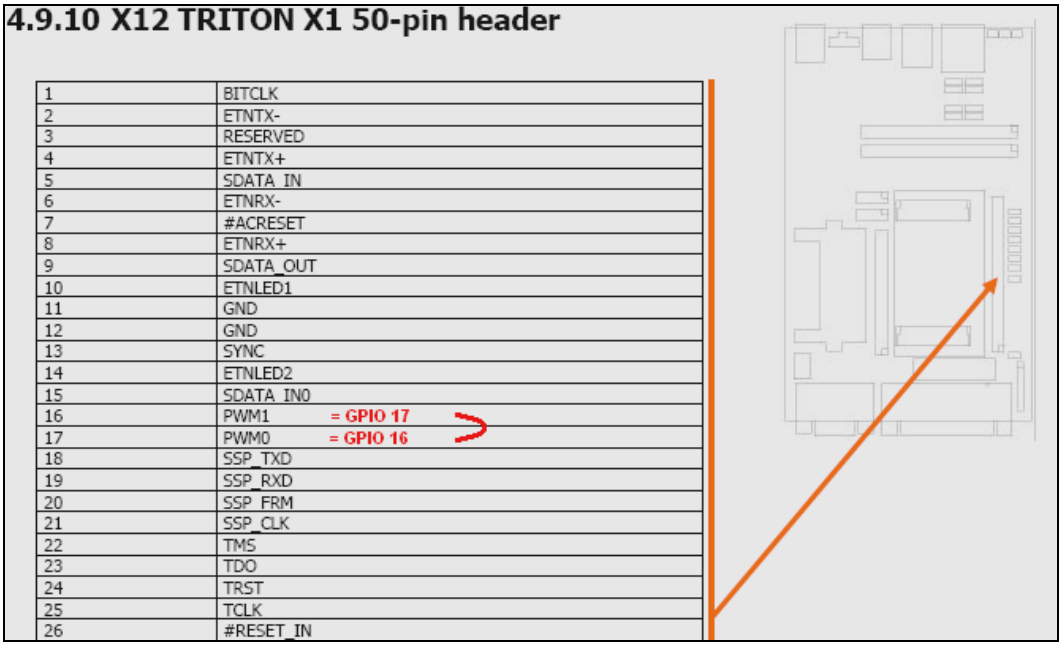


Figure 7-9: X12 connector

Now open two Telnet sessions to the target. With one we will work on GPIO16 and with the other on GPIO17.

Telnet1: Configure GPIO16 as output and a default value of 0

Telnet2: Configure GPIO17 as a rising edge detect interrupt

Telnet2: Wait for interrupts on GPIO17

Telnet1: Set GPIO16 to 1 → an interrupt should be reported on Telnet2

Telnet1: Set GPIO16 to 0

Telnet1: Set GPIO16 to 1 → an interrupt should be reported on Telnet2

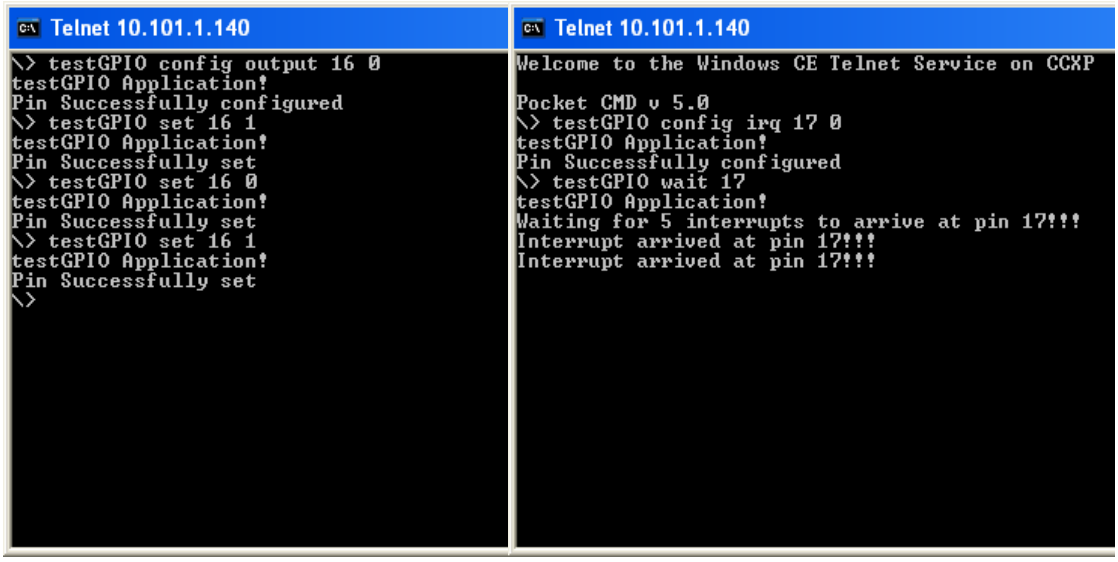


Figure 7-10: Telnet sessions 1 and 2

8. Interfaces & Drivers

8.1. Display

The LCD display driver has been tested with the LQ57Q3DC2 display.

In order to have this component included, you must select *Display support* when you create a new custom platform project or add it later manually to your project.

8.1.1. Modifying the display driver

If you want to use another LCD on the ConnectCore XP module modify the display driver XllpLCDInit() function and make the corresponding changes in *platform.reg*. The datasheet of the corresponding LCD should contain all the necessary values you need to properly set-up the display driver.

8.2. Ethernet

In order to have the component included, you must add to your platform the following component:

- *Wired Local Area Network (802.3, 802.5)*

To delete it from your image, right click over the component group *Network* and select *Delete*. Or you can also click on *Settings…* and then *Exclude from build and Image.*

The TCP/IP configuration expects the system running without DHCP server. The IP address is configured by your U-Boot settings. A special driver (IP2REG) will get that information and write it into the registry before the NDIS driver starts.

To change the default IP, gateway, subnet mask, and DNS of your target you can either change these values in U-Boot or you can disable the IP2REG driver and modify the following key values in *%_WINCEROOT%\Platform\CCXP\Files\Platform.reg*:

```
;Settings for static IP configuration, if enabled
[HKEY_LOCAL_MACHINE\Comm\LAN91C111_1\Parms\TcpIp]
   "EnableDHCP"=dword:0
   "DefaultGateway"="0.0.0.0"
   "UseZeroBroadcast"=dword:0
   "IpAddress"="0.0.0.0"
   "Subnetmask"="0.0.0.0"
   "DNS"="0.0.0.0"
```

If you have a DHCP service and prefer that your target is assigned an IP address from the DHCP server, you must do the following: set the registry configuration *EnableDHCP=dword:1*. Then recompile the platform and download the new kernel. When the target boots, it will take a free IP from the DHCP server automatically.

8.3. USB Host

The USB Host can be included by selecting USB Host support from the catalog and by setting the environment variable BSP_NOUSB to nothing and BSP_USBH_USE=1.

In the catalog, there are different device driver classes, e.g. mouse, keyboard or mass storage can be individually included into the kernel.

The USB Host cannot work with the USB Device at the same time on the development board.

8.4. USB Device

USB Device is added when USB Device Function is added to the project A USB Function Class needs to be added to the project to define the behavior of the USB Device. Windows CE currently supports three classes: Serial, Mass Storage, and RNDIS.

The USB Device cannot work with the USB Host at the same time in the Starter Kit 2 development board.

8.5. Touch screen

The touch screen driver has one environment variable BSP_NOTOUCH which is set to nothing. To exclude the touch driver registry settings set BSP_NOTOUCH=1.

The driver uses the internal SPI channel A to communicate with the ADS7483 Touch controller. When the touch is used, this serial port cannot be used for other purposes.

8.6. FlashFX®

FlashFX® offers the possibility to use the onboard Flash as a normal drive. The driver permits to store files and directories in FAT file system.

FlashFX® includes a Simple Embedded File System (SEFS) that provides an ANSI C style buffered stream interface to files stored on FAT12 and FAT16 drives. By overlaying Datalight's Variable Block Format (VBF) layer, SEFS can access data stored on a FlashFX® disk before either the FlashFX® block device driver or the Windows CE FAT File System are loaded.

In order to have the component included, you must add to your platform the following component:

- *FAT File System*

To delete it from your image, right click the component group *FlashDisk* and select *Delete*. Or you can also click *Settings…* select the *FlashDisk* and then *Exclude from build and Image.*

The partition configuration of the FlashFX® in the registry can be changed by setting the following variables in U-Boot:

- ffxaddr = the partition start
- ffxsize = the partition size

The values should have the syntax <value>MB or <value>KB; the letters can be lower or upper case.

8.7. Hive-based Registry

The hive-based registry stores registry data inside files, or hives, which can be kept on any file system. Each file or hive contains a collection of registry data. The hive-based registry is split into two hives: the system hive, which contains all system data, and the user hive, which contains all data for one particular user. For more information, check the Platform Builder online help.

FlashFX® Pro supports the hive-based registry system, meaning you only need to add the Hive-based Registry component to your kernel.

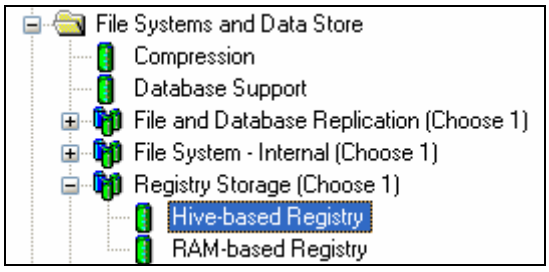


Figure 8-1: Hive-based Registry

The corresponding registry settings are located in *Platform.reg*.

To save changes into the registry on the onboard Flash you can either launch the *savereg.exe* application or select from the start menu of the target device the *Suspend* entry.

8.8. PCCARD

To use the PCMCIA slot on the development board, open the IDE jumper on the development board. If the jumper is closed, the slot is in IDE mode and the PCCARD driver will not work.

The PCCARD driver contains only the hardware of the standard Windows CE PCCARD driver. It doesn't automatically include all necessary drivers to access a memory card, for example. If you want support for a memory, wireless, or any other type of card, you must include the drivers. these the online help for more information of the different PCCARD drivers that are supported by Windows CE and how to include them.

To disable support for the PCCARD, set the variable BSP_NOPCCARD=1. This will remove the driver from the kernel and registry.

8.9. IP2REG

The IP2REG driver configures the MAC address, IP address, Gateway, and DNS of the NDIS driver, as well as the start address of the FlashFX partition and the length, with information extracted from the boot loader (U-Boot environment variables). The interface between U-Boot and IP2REG is shown in the following table:

Environment variable	Description
ipaddr	IP address of target device
gatewayip	Gateway address of target device
netmask	netmask of target device
ethaddr	MAC address of target device
dns	DNS Server IP for target device
ffxaddr	FlashFX partition start address either in kb or MB
ffxsize	FlashFX partition length either in kb or MB

Table 8-2: U-Boot/IP2REG interface

To remove the driver entries from the registry, set the variable BSP_IP2REG to nothing. This will remove the registry entry and result in the driver not being loaded. In this case, all the parameters seen above are extracted from the PLATFROM.REG registry entries.

8.10. GPIO

The GPIO driver implemented here allows the user to easily configure and use GPIO functionality. It allows configuring a GPIO pin to read inputs, set outputs, establish wait methods to signal interrupts, or establish a desired detection method for IRQ (edge/level, high/low…). The driver is not intended to be used in place of other drivers where the performance is critical. Its designed for a user application where several GPIOs have to be handled and the user doesn't want to modify the BSP's OAL.

Once loaded it will take the prefix PIO1:

Several applications can open the driver at the same time since there are synchronization methods to avoid failures.

When developing an application that uses the GPIO driver, you'll need to call GPIO IOCTLs and use the structure defined in the header file:*%_TARGETPLATROOT%\src\inc\BSP_DDK\gpio.h*

The system variable %_TARGETPLATROOT% points to your platform root directory, usually C:\WINCE500\PLATFORM\CCXP

Therefore, if you want to control the GPIOs, you must include this header file in the source code of your application, as shown below:

```
#include <winioctl.h>
#include <BSP_DDK\gpio.h>
```

If you are developing outside Platform Builder, make sure that you export this file together with the SDK. See chapter 7.1 for more details.

The structure used for all GPIO operations is called *GPIOMessage* and has the following fields:

Field	Description
unsigned int unPinNumber	Pin number we want to configure/read/write/wait
GPIOMODE mode	Desired mode for pin when configuring. Must be: GPIO_UNCONFIG, GPIO_INPUT, GPIO_OUTPUT, GPIO_IRQ.
BOOL Block	While configuring, we can block a pin so nobody can change it anymore.
unsigned long ulFlags	Desired flags when configuring a pin as IRQ.
unsigned int unValue	Desired value for outputs. Read value for inputs.

Table 8-3: GPIOMessage structure

The following table lists the IOCTLs used to communicate with the GPIO driver (all use the structure above as argument):

IOCTL	Description
IOCTL_GPIO_CONFIG	Configures a pin
IOCTL_GPIO_WRITE (same as WriteFile function)	Sets value of outputs.
IOCTL_GPIO_READ (same as ReadFile function)	Reads value of inputs and outputs (also pins configured as irq)
IOCTL_GPIO_WAIT_FOR_IRQ	The IOCTL doesn't return until an interrupt arrives to that pin.

Table 8-4: GPIO IOCTL functions

An application called *testGPIO* has been developed to test the driver. See chapter 7.3.1 for information on how to use it and have a look at the code to learn more about the driver.

Be careful when configuring the GPIOs. Most drivers (Ethernet, display, etc) make use of GPIO pins. If you change the default configuration of a pin that is used by a driver, it may stop working properly.

The GPIO driver is normally included when you create a project. If you want to exclude it set the environment variable BSP_NOGPIO to 1 and rebuild the BSP.

9. Tips & Tricks

9.1. Autostart applications

Here is an extract of file *COMMON.REG*:

```
[HKEY_LOCAL_MACHINE\init]
; @CESYSGEN IF CE_MODULES_SHELL
        "Launch10"="shell.exe"
; @CESYSGEN ENDIF
IF IMGTINY !
; @CESYSGEN IF CE_MODULES_DEVICE
        "Launch20"="device.exe"
; @CESYSGEN ENDIF
; @CESYSGEN IF CE_MODULES_GWES
```

In order to launch your own applications automatically after boot of the Windows CE kernel, you need to add similar entries to the file *PROJECT.REG*. You can also create brand new variables that let you control whether or not to launch your applications as in the following example:

```
IF MY_VARIABLE1
        “Launch40”=“my_application1.exe”
ENDIF
IF MY_VARIABLE2
        “Launch50”=“my_application2.exe”
ENDIF
```

9.2. Telnet server

To connect via Telnet simply open a DOS box and telnet to the IP address of your target. You won’t be requested to enter a username and password, because User Authentication is disabled by default. To exit the Telnet server, simply enter `exit` at the prompt.

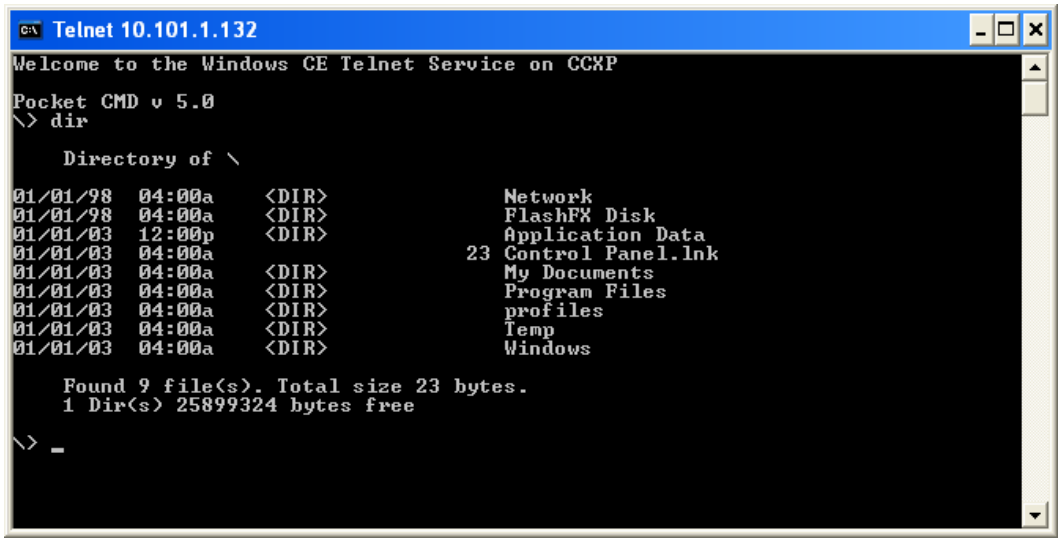


Figure 9-1: Telnet session in the target

9.3. FTP server

To connect via FTP to the FTP server of the target, open a DOS box and simply do an FTP to your target IP address. The target will ask you for a username and a password. We have enabled the anonymous login; enter anything as username and anything as password (you must enter something when using the DOS box).

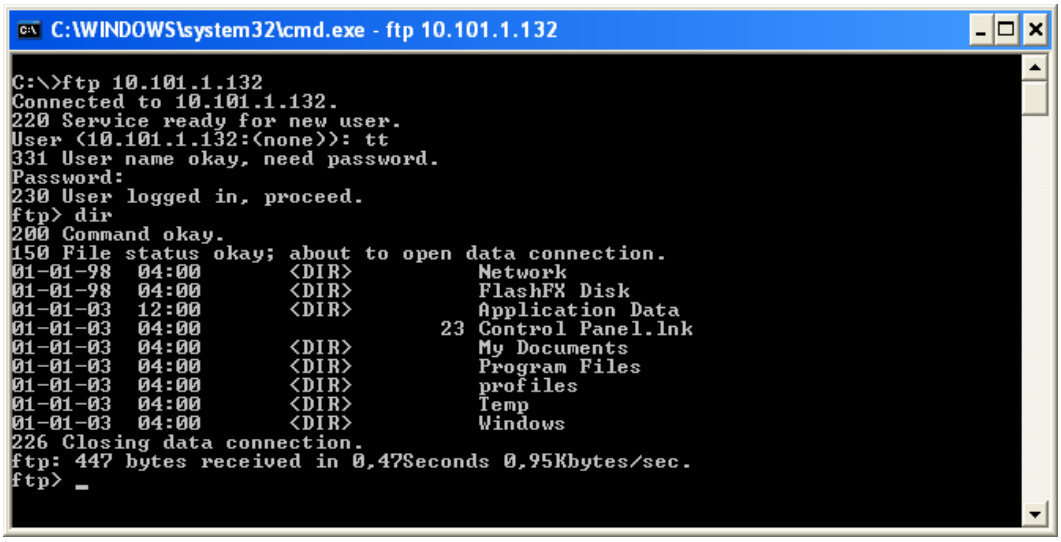


Figure 9-2: FTP session in the target

To finish the FTP connection, simply type the word bye at the FTP prompt.

You can also connect using Microsoft Internet Explorer. In this case you won't be asked to introduce a login or password.

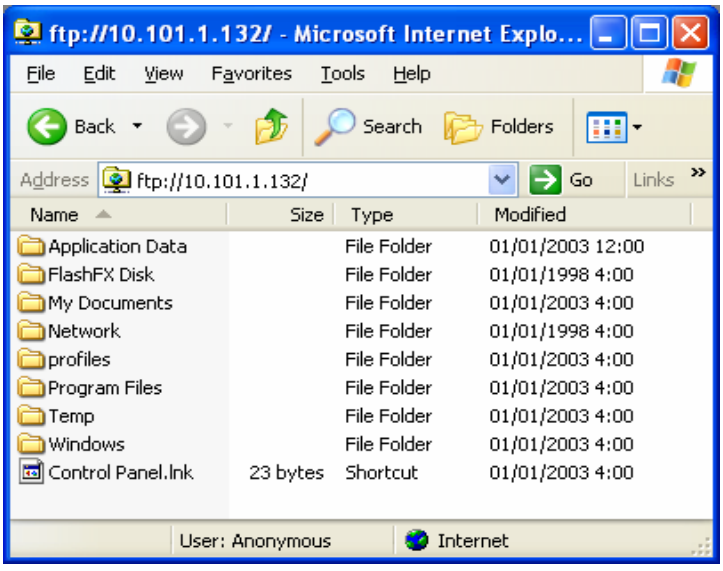


Figure 9-3: FTP session from Internet Explorer

The FTP sessions from a DOS box have problems with folder names that include blank spaces like "My Documents". To avoid this problem, use Microsoft Internet Explorer which can handle names with blank spaces.

10. Troubleshooting

10.1. Removing the BSP

In order to completely remove the BSP, follow these instructions:

- Delete the CE Components file *%_WINCEROOT%\public\common\oak\catalog\cec\CCXP.cec*
- Go to the *File* menu of Platform Builder and click on *Manage Catalog Items*

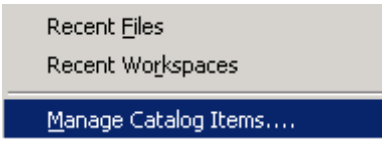


Figure 10-1: Manage Catalog Items

- Select the platform to remove and click the *Remove* button.

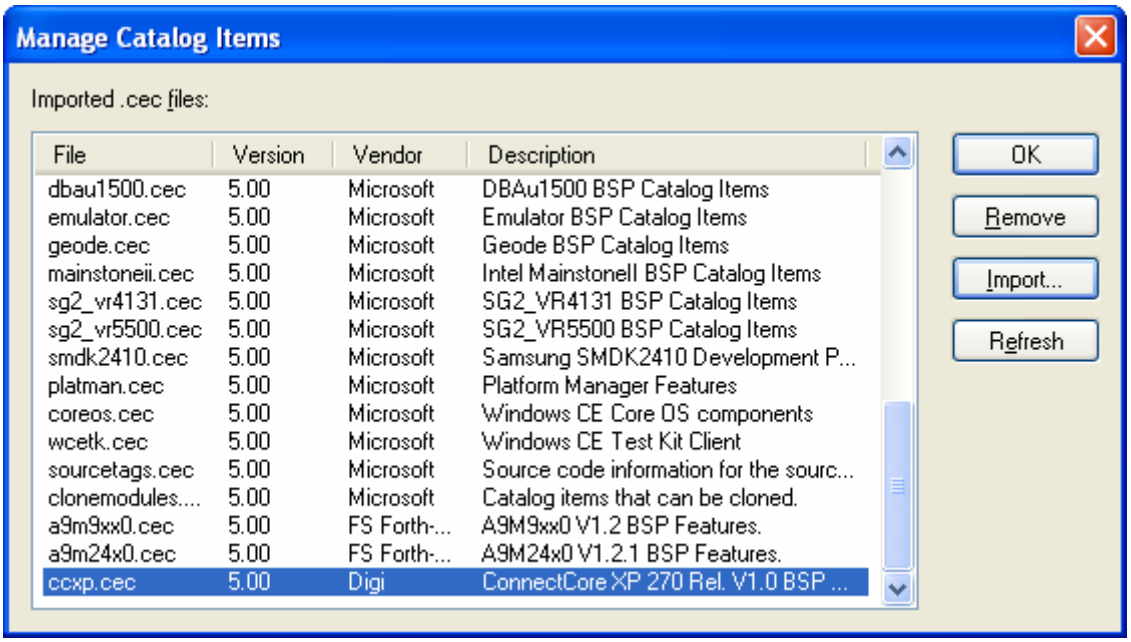


Figure 10-2: Removing the BSP

- Delete the directory *%_WINCEROOT%\PLATFORM\CCXP* and subfolders.
- Delete the directory *%_WINCEROOT%\PLATFORM\COMMON\SRC\ARM\INTEL\CCXP* and subfolders.

10.2. Language settings

When using different languages an error may occur during the build process because the default language for the CCXP package is English. This is important if you want to add you own application to the Kernel. Some folders have different names depending on the selected language (for example *My Documents* changes to *Mis Documentos* in Spanish language). Because of the

language you may not see your application in the target. This occurs because the path is incorrect. To use your application in different languages, use the *.dat files inside your project. Here you can specify the name of the folder depending of the used language. For more information please refer to Platform Builder online Help.

10.3. Quick Fix Engineering (QFE)

Microsoft releases periodic updates of the Platform Builder and the source codes. This package already includes some QFEs. In case of kernel misbehavior, check the Microsoft web side for new QFE releases.

http://msdn.microsoft.com/embedded/downloads/ce.net/wince/default.aspx

Each QFE contains a readme file that lists all the updates that have been made with the new installation. We recommend always installing all new QFEs.

Installed QFEs packages can be checked with the tool *CEQFECheck.exe* which can be found in *%SystemRoot%\System32\ceqfecheck*

11. Appendix A

11.1. CD contents

The supplied CD contains all the software and documentation you need to start your evaluation of Windows CE 5.0 on one of the CCXP platforms.

Folder/File	Description
Apps	Test applications, like testGPIO
Bootloader	Boot loader binaries
Doc	Documentation
Images	Windows CE kernel binaries and script files
QFEPacks	Microsoft Quick Fix Engineering software updates for Windows CE 5.0
SDK	Contains the Software Development Kit for a sample platform.
U-Boot	Boot loader sources
Cygwin	Linux environment for Windows that contains Microcross X-tools tool chain
BSP	Contains the ConnectCore XP 270 BSP CCXP.msi
Setup.exe	Installation program

Table 11-1: Contents of the CD

11.2. Windows CE directory tree

Here is an overview of the most important directories and files in the Windows CE tree.

11.2.1. Main directories

Directory	Description
%_WINCEROOT%\PLATFORM	In this folder are the board specific modules. For example: CEPC Microsoft reference PC platform GEODE Microsoft reference GEODE platform CCXP ConnectCore XP 270 platform
%_WINCEROOT%\PBWorkspaces	All project specific parts are stored here, for example: MyCCXP One CCXP based project. test_CEPC One CEPC based project
%_WINCEROOT%\PUBLIC\COMMON	In this directory all common adjustments and drivers are stored. Be careful with changes in this directory and subfolders. A change in one of the sources in the common directory will have repercussions in all platforms.

Figure 11-2: Main Windows CE directories

11.2.2. BSP Component file

With the CEC Editor included in Windows CE 5.00, you can explore the components of the BSP by double-clicking the file *%_WINCEROOT%\public\common\oak\catalog\cec\CCXP.cec*

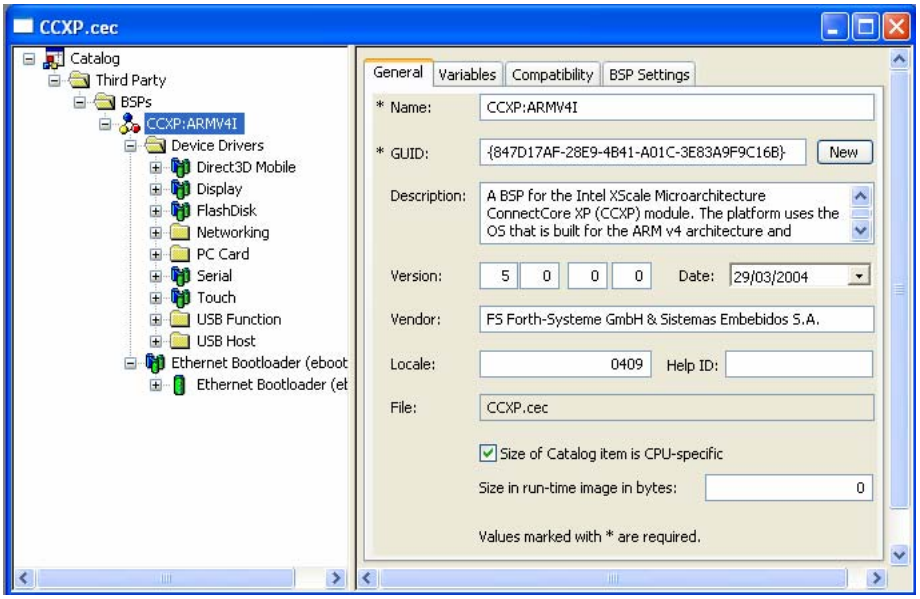


Figure 11-3: BSP Component file

This package includes the following components:

- LCD display driver
- Ethernet driver
- Serial port
- USB Host
- USB Device
- Touch Screen
- FlashFX®
- PCCARD
- IP2REG

11.2.3. Other important files

File	Description
SOURCES	Contains source files, include/library files and the names of the output files.
DIRS	Subfolders that should be compiled
*.reg	Windows CE registry files

Table 11-4: Important Files

11.3. Flash layout

The following table describes the Flash layout of the ConnectCore XP 270 module:

Flash start address	Flash end address	Purpose
0x00000000	0x0003FFFF	Bootloader (U-Boot)
0x00040000	0x0007FFFF	Bootloader configuration
0x00080000	0x000BFFFF	Bootloader
0x00100000	0x01FBFFFF	Windows CE Kernel and FlashFX partition
0x01FC0000	0x01FFFFFF	U-Boot splash image

Table 11-5: Flash layout

11.4. Links

These links represent the current status of this documentation at the time of release and may change without notice

http://www.digi.com Manufacturer of ConnectCore XP 270 module

http://www.embebidos.com Windows CE integrator

12. Appendix B

12.1. U-Boot command reference

This chapter gives an overview of common used U-Boot commands.

More detailed information can be found at:

http://www.denx.de/twiki/publish/DULG/DULG-tqm8xxl.html#Section_1

To get to the U-Boot prompt press any key immediately after you have powered the target on or pressed reset. At the prompt type “help” or “?” to get an overview of the supported commands.

```
CCXP270> help
?       - alias for 'help'
autoscr - run script from memory
base    - print or set address offset
bdinfo  - print Board Info structure
boot    - boot default, i.e., run 'bootcmd'
bootd   - boot default, i.e., run 'bootcmd'
bootm   - boot application image from memory
bootp   - boot image via network using BootP/TFTP protocol
clock   - Set Processor Clock
cls     - clear screen
cmp     - memory compare
coninfo - print console devices and information
cp      - memory copy
crc32   - checksum calculation
echo    - echo args to console
erase   - erase FLASH memory
fatinfo - print information about filesystem
fatload - load binary file from a dos filesystem
fatls   - list files in a directory (default /)
flinfo  - print FLASH memory information
go      - start application at address 'addr'
help    - print online help
iminfo  - print header information for application image
imls    - list all images found in flash
itest   - return true/false on integer compare
loadb   - load binary file over serial line (kermit mode)
loads   - load S-Record file over serial line
loop    - infinite loop on address range
md      - memory display
mm      - memory modify (auto-incrementing)
mtest   - simple RAM test
mw      - memory write (fill)
nfs     - boot image via network using NFS protocol
nm      - memory modify (constant address)
ping    - send ICMP ECHO_REQUEST to network host
printenv- print environment variables
protect - enable or disable FLASH write protection
rarpboot- boot image via networkautoscr - run script from memory
```

Each of these commands has additional help available, which can be viewed by entering help <command>.

All numeric values, which are needed for different commands, are interpreted as HEX values. Entering 30100000 means 0x30100000.To speed up programming, the real size of the image files can be used.

The following table explains some of the more often used commands:

Command	Description
bootm ADDR ARG	boots image from ADDR passing arguments ARG. ARG is the address of the initrd image
boot, bootd	boots image via running default bootcmd
erase START END/+SIZE	erase from START to END / erase from START +SIZE
cp source target count	copies count [b,w,l] from source to target
printenv	prints the environment variables
saveenv	stores the changed environment variables persistently
setenv VARIABLE VALUE	sets the environment variable VARIABLE to the given value VALUE. If a semicolon is used, to set different variables, it has to be masked with "\"
run VARIABLE	executes the commands of VARIABLE like a script
tftp ADDR image	loads image to ADDR via network using TFTP and the environment variables "ipaddr" and "serverip"
usb reset	enables and resets the USB interface
usb scan	scans the bus for attached USB storage devices
usb tree	shows the connected devices
fatload usb DEV:PART ADDR image	loads image to ADDR from USB storage device DEV with the partition number PART to ADDR
help	shows all of the available commands
help ITEM	shows all of the available commands belonging to a particular item. e.g. help nand
bootm ADDR ARG	boots image from ADDR passing arguments ARG. ARG is the address of the initrd image
boot, bootd	boots image via running default bootcmd
erase START END/+SIZE	erase from START to END erase from START +SIZE
cp source target count	copies count [b,w,l] from source to target
printenv	prints the environment variables
saveenv	stores the changed environment variables persistently
setenv VARIABLE VALUE	sets the environment variable VARIABLE to the given value VALUE. If a semicolon is used, to set different variables, it has to be masked with "\"
run VARIABLE	executes the commands of VARIABLE like a script
tftp ADDR image	loads image to ADDR via network using TFTP and the environment variables "ipaddr" and "serverip"
usb reset	enables and resets the USB interface
usb scan	scans the bus for attached USB storage devices
usb tree	shows the connected devices
fatload usb DEV:PART ADDR image	loads image to ADDR from USB storage device DEV with the partition number PART to ADDR
help	shows all of the available commands
help ITEM	shows all of the available commands belonging to a particular item. e.g. help nand

Table 12-1: Common U-Boot commands

12.2. Building U-Boot

If you want to enhance the boot loader or reduce the size of the current boot loader, you can build your own customized U-Boot. To build a new U-Boot version, you must install the Microcross Cygwin X-tools and the U-Boot sources from the CD.

After successful installation, open the Cygwin build environment. Move to the directory where you have previously installed the U-Boot sources (the default is c:\u-boot) and type the following command:

```
$   ./userbuild_cyg.sh ccxp270stk2
```

This will build a new boot loader and create a new image called *u-boot-ccxp270stk2.bin* in the root directory of the U-Boot source folder. When the compilation has successfully terminated, update the new boot loader on your target as stated in chapter 4.4.