Name:_____ **SID:** _____

CSCE 351: Operating System Kernels

Lab 1 – Introduction to Windows CE .NET Platform Builder

Basic Setup:

- Windows 2000/XP workstation with Windows CE .Net 4.2 installed.

Objectives:

The objectives of this lab are as follows:

- Familiarize students with the development process in platform builder.
- Expose students to the basic debugging process in platform builder.
- Illustrate how to download a design to a device that can be either an actual system (e.g. E-box) or the provided emulator.
- Exploring some of the remote monitoring and debugging capability in platform builder 4.2.

Estimated Lab Time: 60 minutes

Introduction

Microsoft Platform Builder is an integrated development environment (IDE) for building customized embedded platforms based on the Microsoft Windows CE .NET operating system (OS). Platform Builder comes with all the development tools necessary for you to design, create, build, test, and debug a Windows CE–based platform. The IDE provides a single integrated workspace in which you can work on both platforms and projects.

As a student in CSE, you are able to download a copy of Windows CE .Net 4.2 with academic license. The file is available from **http://msdn08.e-academy.com/unl_cseng.** In order to download, you need to send an email to **manager@cse.unl.edu** to set up an account. Please specify that you need to download Windows CE .Net 4.2 and you are a student in CSCE 351. You should try to install Windows CE .Net 4.2 on your personal machine as well.

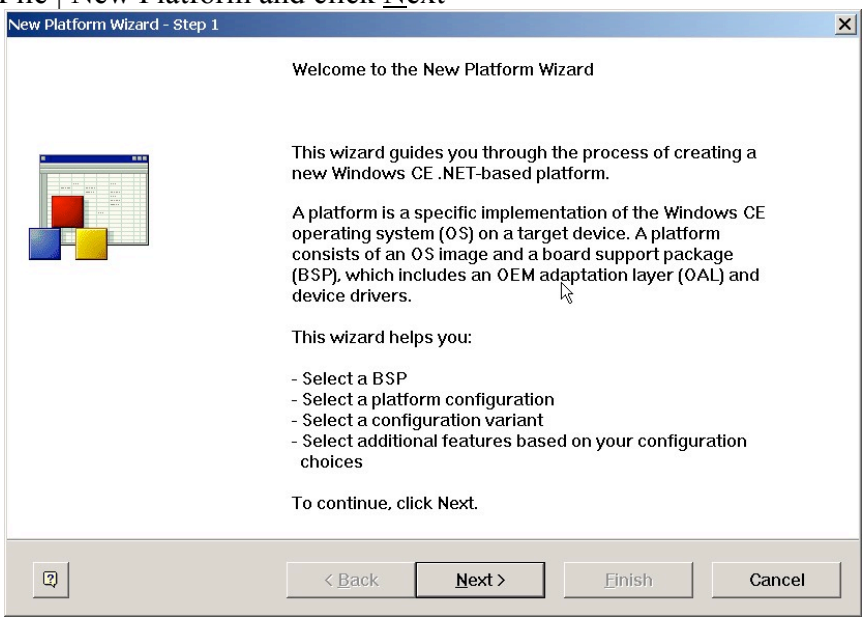
In this lab, we will learn how to create a platform, add/delete component to the platform, compile and debug the platform, download image to the device (the built-in emulator).

1. Platform Creation with the New Platform Wizard

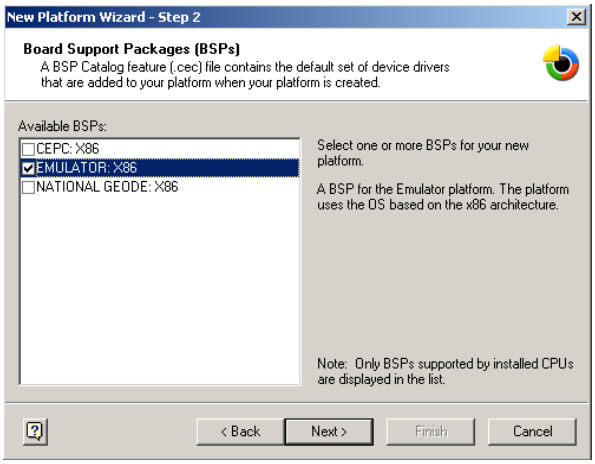
Step 1. Open platform builder

Step 2. Once launched, you need to create a new platform workspace. To do so, you select:

- File | New Platform and click Next



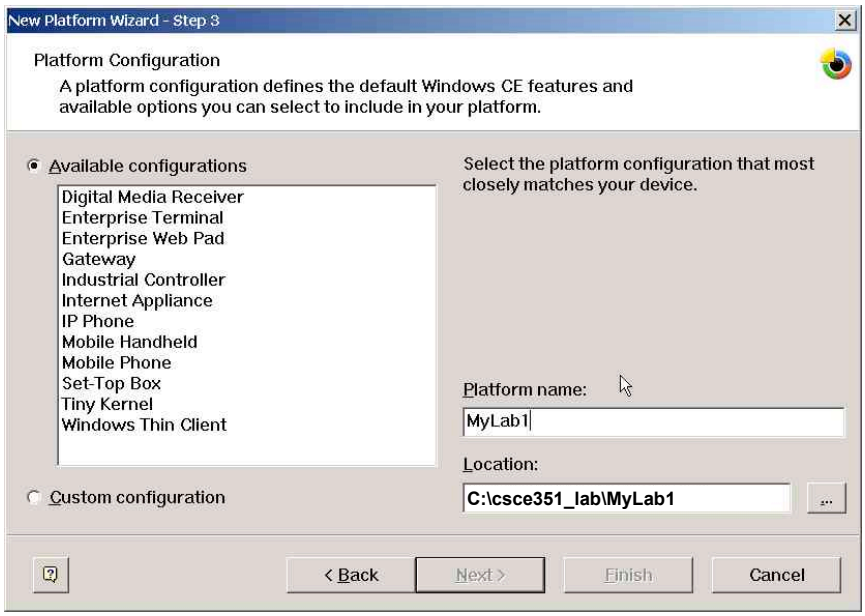
- Check EMULATOR: X86 and click Next. **Notice:** Windows CE .NET provides an Emulator, which is a tool that mimics the behavior of hardware that supports a Microsoft Windows CE–based platform. With the Emulator, you can design and build a Windows CE–based platform and test it using software that mimics hardware rather than testing the platform on a real system.



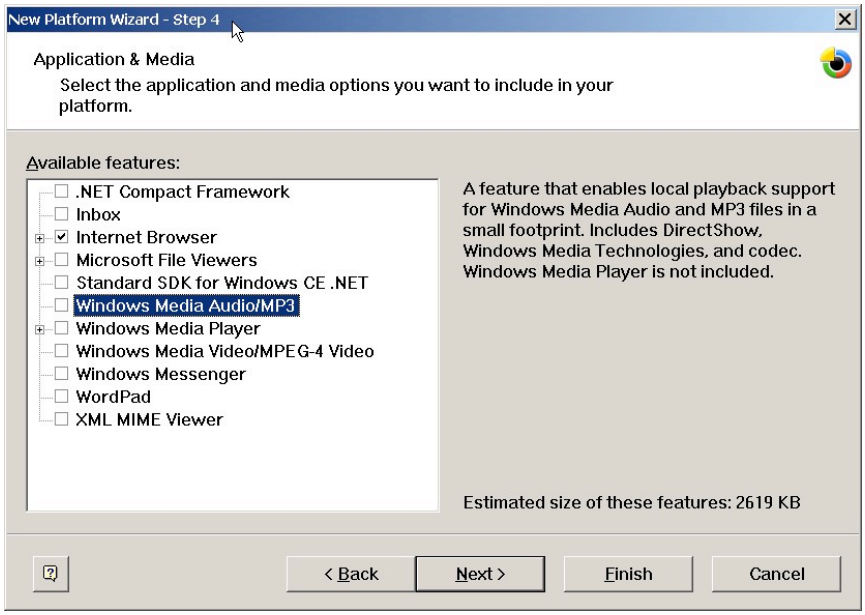
Question: the list represents various embedded processors that the tool can support. As part of the post lab, please provide a complete specification for CEPC: X86 that include the most current processor clock rate and a sample device that uses that particular processor.

- Select Internet Appliance from a list of Available configurations, input platform name as MyLab1, and click Next.

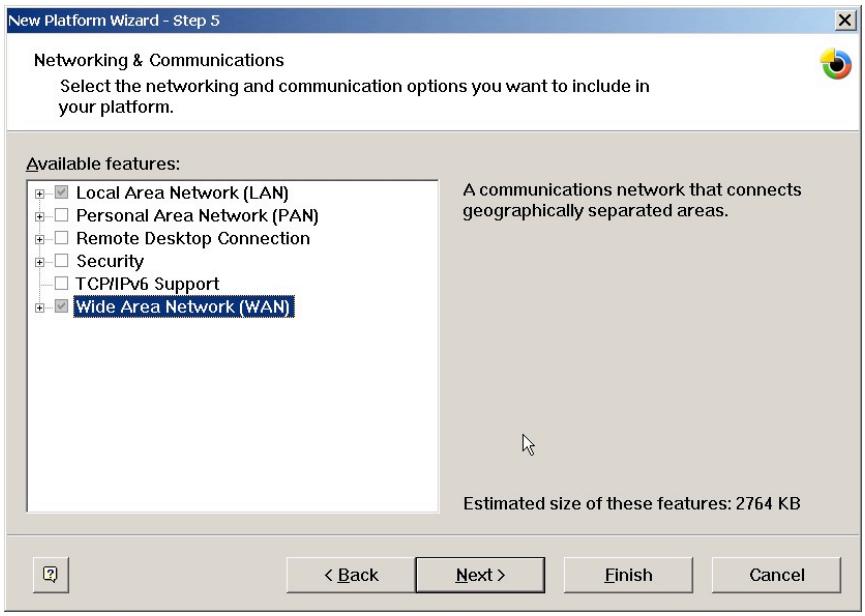
Question: Notice, there are several types of devices that are supported by Windows CE .NET. As part of your post lab, answer the following questions: i) What is a set top box? ii) List one manufacturer of set-top-box that uses CE .NET as the operating system.



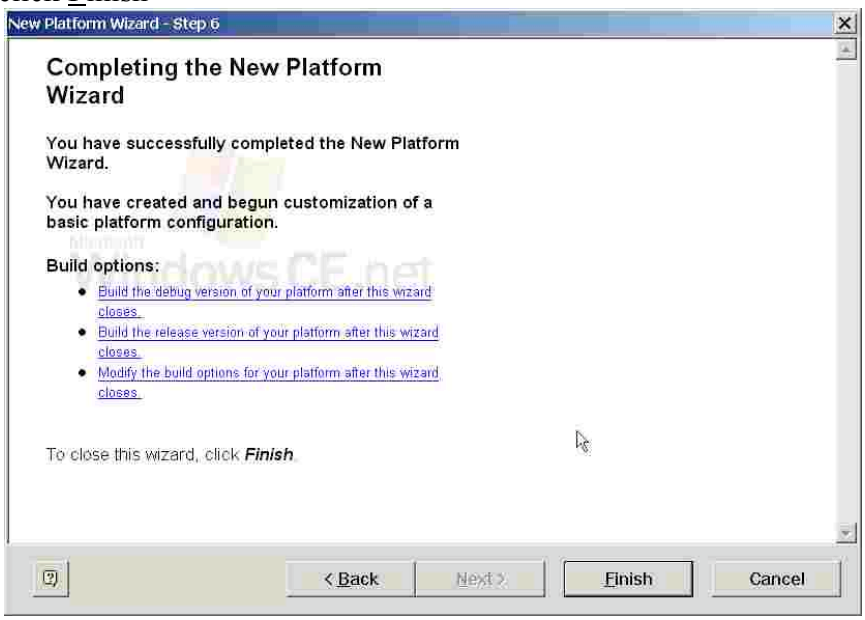
- Make sure that Internet Brower is checked in the list available features and other features are not checked, and then click Next.

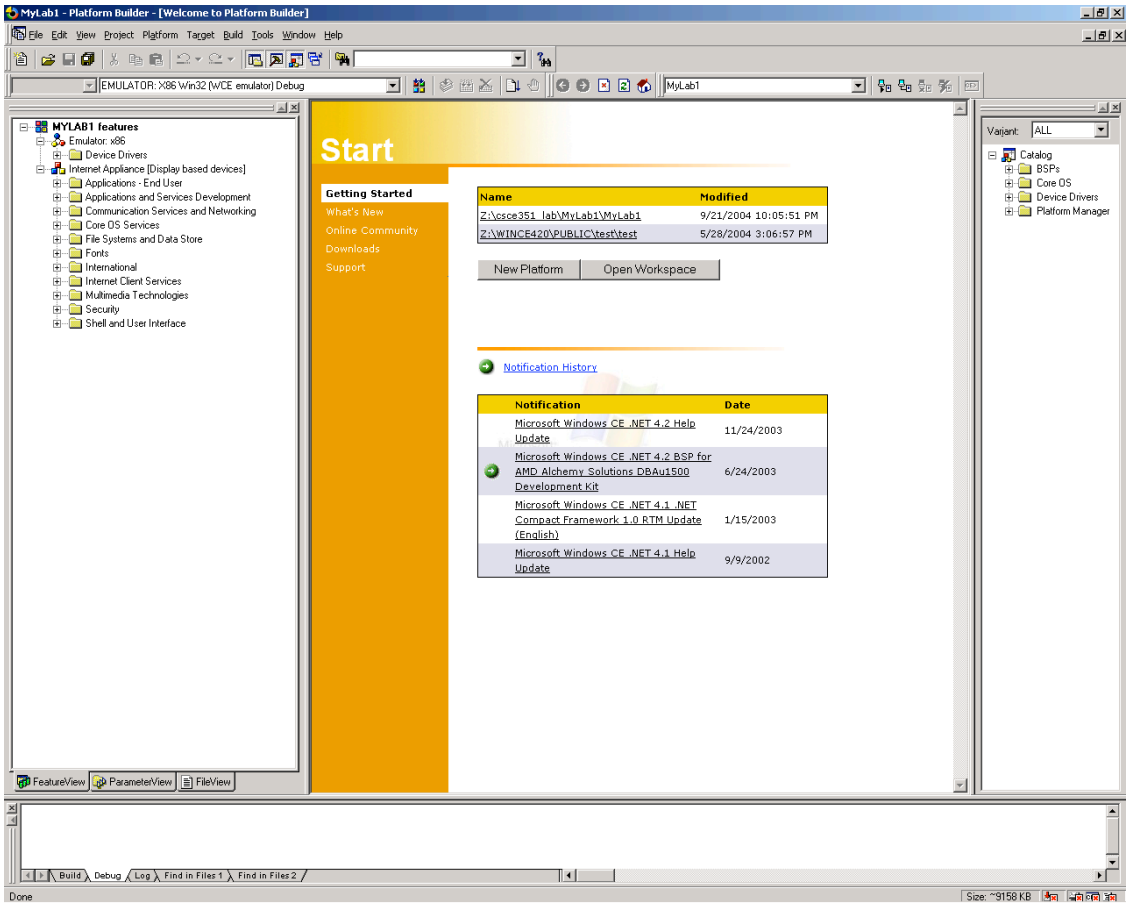


- click Next



- click Finish

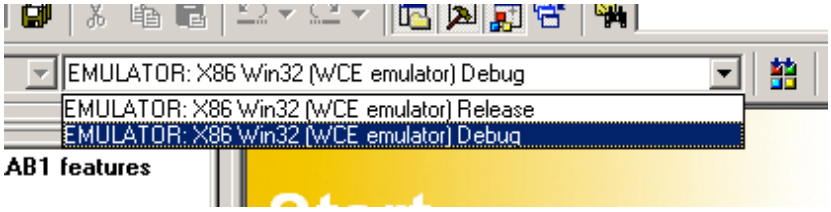




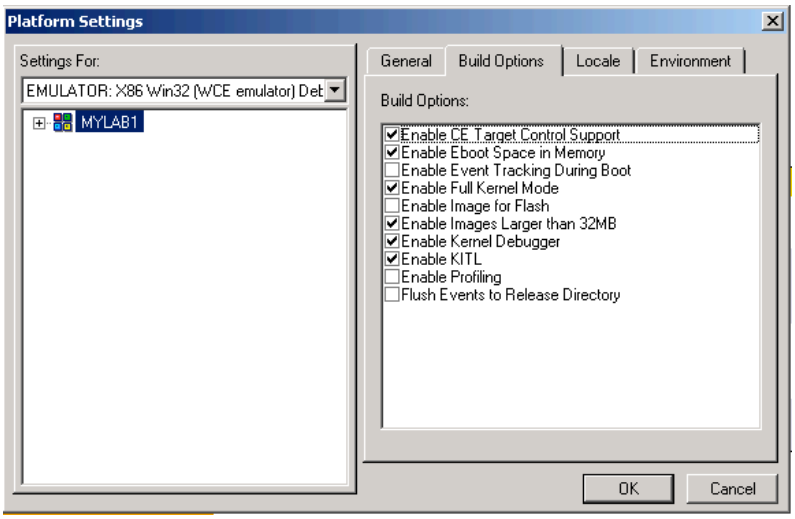
2. Customize and Build the Platform

In this section, we will perform simple modification to the kernel. This step will take about 30 minutes.

Step 1. Switch from Release Build to Debug Build



Step 2. Select Platform | Setting | Build Options and *Enable Kernel Debugger*



Step 3. Select Build | Build Platform

This step takes a bout 15-20 minutes to finish. While you are waiting for the build process to complete, please read Appendix A, *History of Windows CE*.

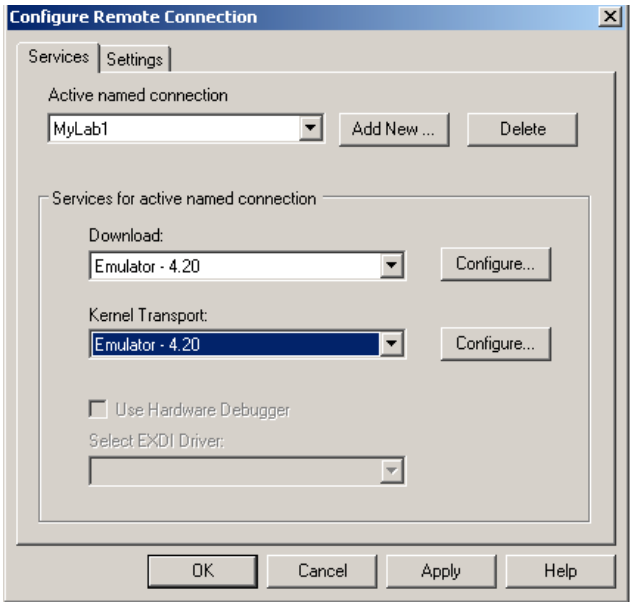
The build system builds a platform in four sequentially executed phases: the Sysgen phase, the Feature Build phase, the Release Copy phase, and the Make Image phase. Please go to this link for more information about build phases:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wcepb40/html/_wcepb_Building_an_OS_Image.asp

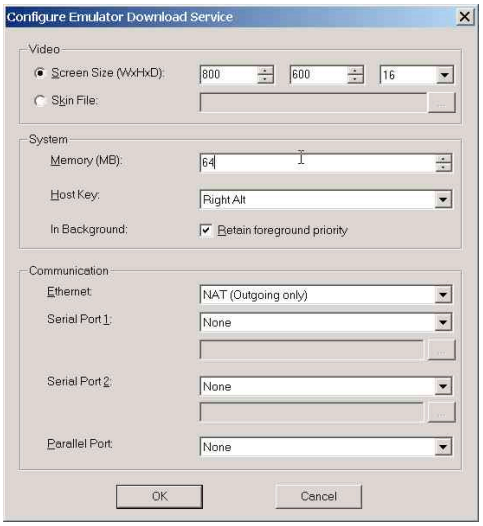
If the build succeeds, the build information should contain 0 errors.

3. Download OS image to the emulator

Step 1. In platform builder, Select Target | Configure Remote Connection
Step 2. Select *Emulator – 4.20* for both *Download* and *Kernel Transport*

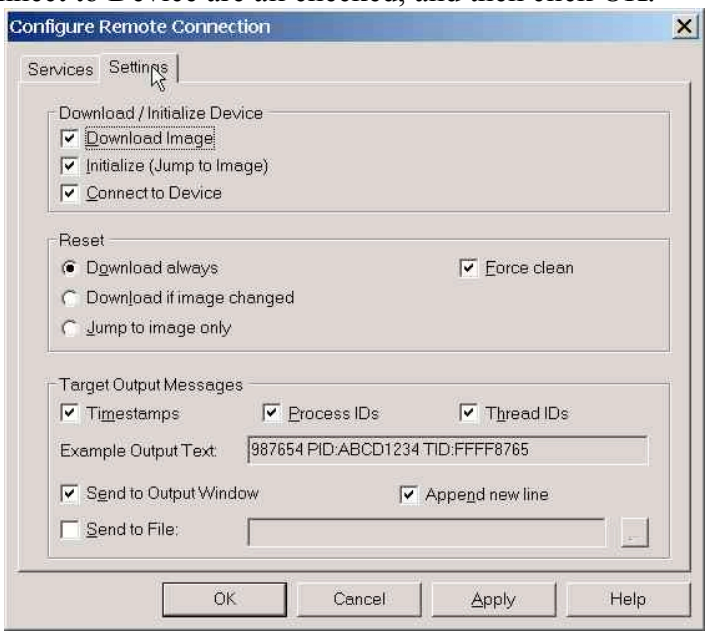


Step 3. Increase the memory size to 64MB



Step 4. Make sure that the NAT is selected for *Download* configure in order to connect to internet

Step 5. Click the setting Tab, and make sure Download Image, Initialize (Jump to Image), and Connect to Device are all checked, and then click OK.

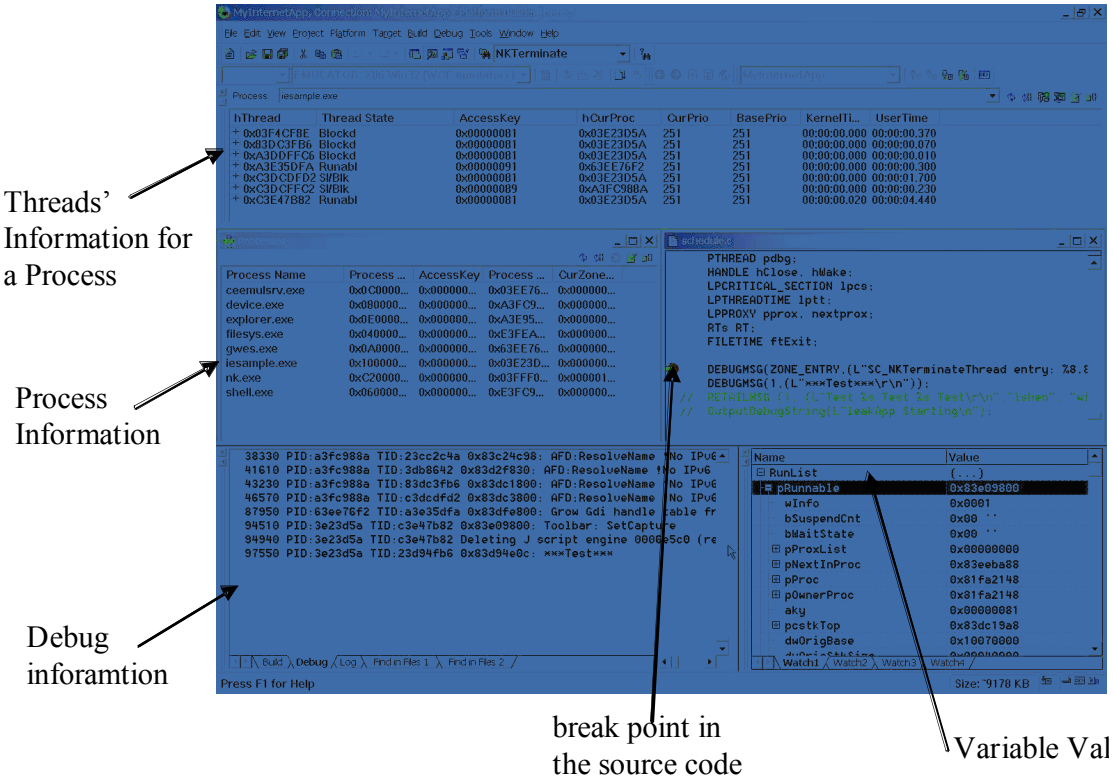


Step 6. Select Target | Download/Initialize

Step 7. After download is finished, the following window will show up in the emulator display (look in your task bar to find the emulator display program).

4. Platform Debugging

We can debug emulator-based platform. After the download is finished, the platform builder enters the debug mode. We can watch the debug message, the process and thread states. We can also set break points at source code and watch the variable values.



- To debug, you can Select what you want to see in View | Debug Windows
- To set break point,
  - Select Debug | Break
  - Open the source code from File | Open
  - Set the break point in the source code
  - Select Debug | Go

We give an example in the following:

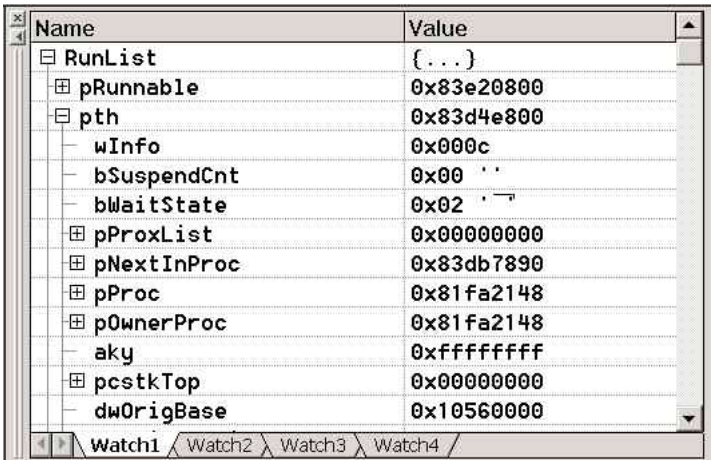
Step 1. Download the image into the Emulator
Step 2. Select View | Debug Windows | Processes
Step 3. Select View | Debug Windows | Variables
Step 4. Open an application in the emulator window, e.g., internet explorer
Step 5. Select Debug | Break
Step 6. In the platform builder, click file | open to open the file %WINCEROOT\PRIVATE\WINCEOS\COREOS\NK\KERNEL\schedule.c
Step 7. Set up a break point in function SC_NKTerminateThread

```
//------------------------------------------------------------------------------
//------------------------------------------------------------------------------
void
SC_NKTerminateThread(
    DWORD dwExitCode
    )
{
    int loop;
    LPCRIT pCrit;
    PCALLSTACK pcstk, pcnextstk;
    LPCLEANEVENT lpce;
    PTHREAD pdbg;
    HANDLE hClose, hWake;
    LPCRITICAL_SECTION lpcs;
    LPTHREADTIME lptt;
    LPPROXY pprox, nextprox;
    RTs RT;
    FILETIME ftExit;

    DEBUGMSG(ZONE_ENTRY,(L"SC_NKTerminateThread entry: %8.8lx\r\n",dwExitCode));

    SETCURKEY(0xffffffff);
    if (GET_DYING(pCurThread))
        dwExitCode = GetUserInfo(hCurThread);
    SET_DEAD(pCurThread);
```

Step 8. Select Debug | Continue.
Step 9. Close an application in the Emulator window, (e.g. Internet Explorer).
Step 10. Now the kernel stops at the breakpoint.
Step 11. In the Variable window, click the cell under Name and input *RunList*.
Step 12. *RunLis*t contains a list of threads, the current running thread is called *pth*. You can open it to see detail information.



Step 13. Feel free to debug other functions and open more watch windows.