**Samraksh 8200.NET Application Development Tutorial**

Author: Mukundan Sridharan and Nived Sivadas  
Revision Date: Feb 02, 2012  
Version: 1.4

**1.0 Soc8200/AM3517/Cortex-A8 NETMF Porting Kit and CLR Development Environment Setup:**

**1.1 Software installation**

* Install c# and c++ visual studio 2008 express edition  
  **Note:** It is important to stick to C# 2008 Express edition, C# 2010 Express Edition does not work for developing Interops
* Install NETMF 4.0 sdk, available at , <http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=23546>
* Install the NETMF 4.0 Porting kit, available at, <http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=24259>

**Note: It is customary to install porting kit under C:\MicroFrameworkPK\_v4\_0. This is not a must though. If you install it somewhere else, you need to manually edit the hardcoded references in the following files**

* Solutions\ Soc8200\ TinyCLR\TinyCLR.proj
* InteropLibrary\NativeCode\dotNetMF.proj
* MyInterOp\MyEncryptionLib\NativeCode\AES\_MyEncryptionLib\_SSL.cpp:14:
* DeviceCode\Targets\Native\TIAM3517\DeviceCode\drivers/crypto/aes\_native.c:11:
* DeviceCode\Targets\Native\TIAM3517DeviceCode\drivers/crypto/ssltest.c:11:
* InteropLibrary\NativeCode\ADC\_library.featureproj
* InteropLibrary\SimpleInterop\NativeCode\SimpleInterop.featureproj
* MyInterOp\MyEncrptionLib\NativeCode\MyEncryptionLib.featureproj
* DeviceCode\Targets\Native\TIAM3517\DeviceCode\drivers\test\_native\_drivers\main.cpp
* Install OpenOCD dev, download 32-bit or 64-bit depending on your system, http://www.freddiechopin.info/index.php/en/download/category/10-openocd-dev
* Install usb drivers for Olimex JTAG Adapter drivers. (This depends on the adapter you use, but usually the below driver works)
  + Get the OpenOCD 0.4.0 driver for Olimex ARM-USB-Tiny-H from the page <http://www.olimex.com/dev/arm-usb-tiny-h.html>, look under 'Software'
  + Direct link to driver, http://www.olimex.com/dev/soft/arm/JTAG/DRIVERS%20(libusb-1.2.2.0,%20CDM20808).zip
  + Unzip the files in drivers directory, plugin the dev-board, go to control panel,

device manager, update driver and select driver directory

* Download C++ toolchain from Code sourcery for ARM EABI for Windows, from

- http://www.codesourcery.com/sgpp/lite/arm/portal/release1802

* Download Samraksh NETMF Porting kit from Samraksh GitHub or get a copy from Samraksh Support Team (This is not opensource)
  + GitHub at (Get link from Samraksh Support Team)
  + Copy the Samraksh porting kit source tree on top of the Microsoft porting it source tree.

**1.2 Setuping up the Environment and Compiling**

* Open CMD Prompt
* Go to folder c:\MicroFramework.. or which ever driver you have installed the porting kit code downloaded.
* Set the environment using the command
  + setenv\_gcc.cmd c:\CodeSourcery
    - where CodeSourcery is the folder where you downloaded the ARM compilers
  + **Note**: If you use Eclipse you should start Eclipse from command line now, after you setup the environment, so that Eclipse can correctly find the compliers.
* Compile the TinyCLR (**RAM build**)
  + Go to folder C:\MicroFrameworkPK\_v4\_0\Solutions\Soc8200\TinyCLR and
  + run 'msbuild /t:clean tinyclr.proj' to clean the source.
  + run 'msbuild /t:build /p:memory=RAM tinyclr.proj' to build the source.

**1.3. Using Eclipse for Devlopment**

This section is not a must. You can use any editor to do your development and use command line to compile. But it is nice to get Eclipse working.

A good tutorial is available at <http://msmicroframework.blogspot.com/2009/02/configuring-eclipse-openocd-gcc-to.html>, just follow that. At this point you are already done with Codesourcery setup, so proceed directly to Zylin Embedded CDT setup.

**Note:** Zylin CDT for Eclipse has moved. It is now available at: http://opensource.zylin.com/embeddedcdt.html

The Eclipse update site is at http://opensource.zylin.com/zylincdt

**1.4 Other useful references:**

1. Basic Interop tutorial: <http://informatix.miloush.net/microframework/Tutorials/Interop.aspx>
2. Eclipse NETMF tutorial: <http://msmicroframework.blogspot.com/2009/02/configuring-eclipse-openocd-gcc-to.html>
3. Beginners Guide to C# and NETMF from GHI Electronics: <http://www.ghielectronics.com/downloads/FEZ/Beginners%20guide%20to%20NETMF.pdf>
4. Deploying from command line: http://www.eggheadcafe.com/microsoft/NET-Microframework/35747437/how-to-deploy-to-netmf-device-from-command-line.aspx

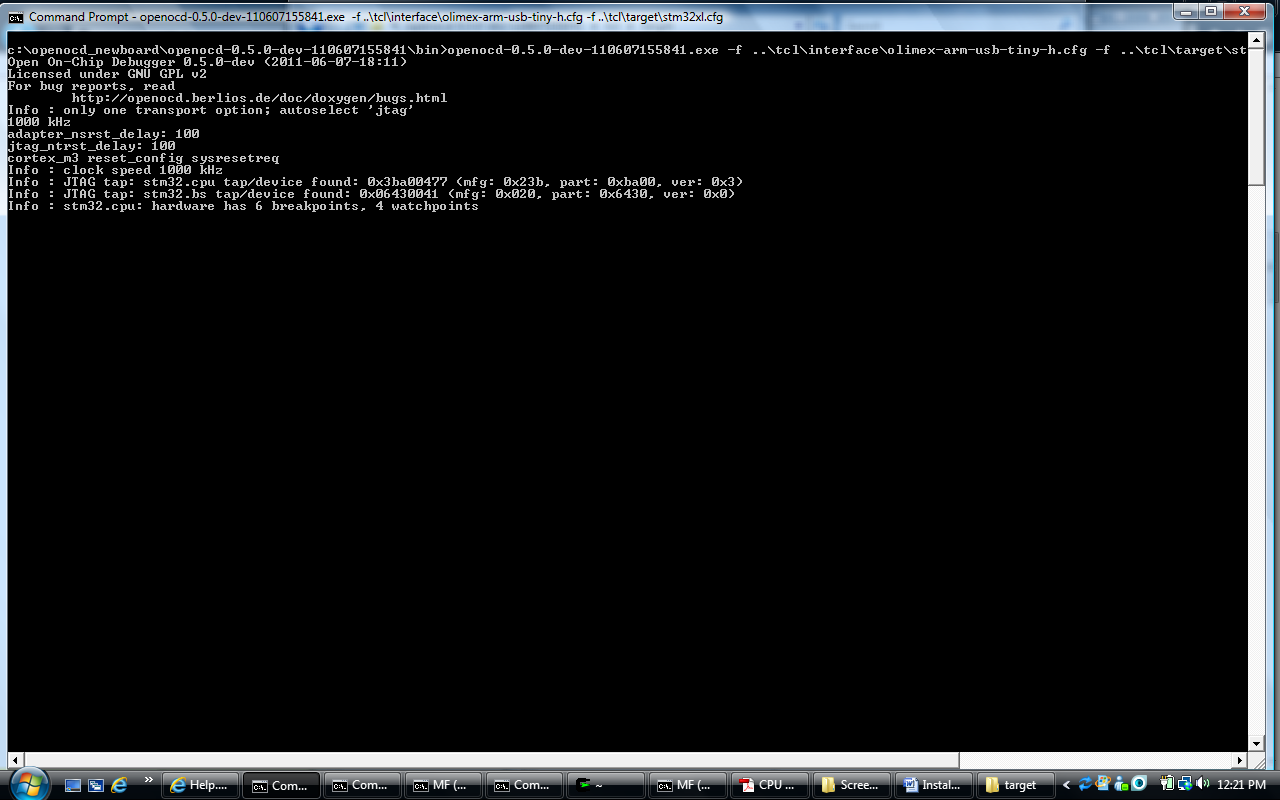
**2.0 Downloading Software to Emotes through JTAG**

1. Download [OpenOCD](http://www.freddiechopin.info/index.php/en/download/category/4-openocd) version 0.5.0 (32 bit or 64 bit). This software drives the JTAG Programmer and enables you to program the emote. The software comes pre compiled and there is no installation process required.

2. Starting OpenOCD with command.

**openocd-<>.exe –f ..\tcl\interface\olimex-arm-usb-tiny-h.cfg –f ..\tcl\boards\am3517evm.cfg**

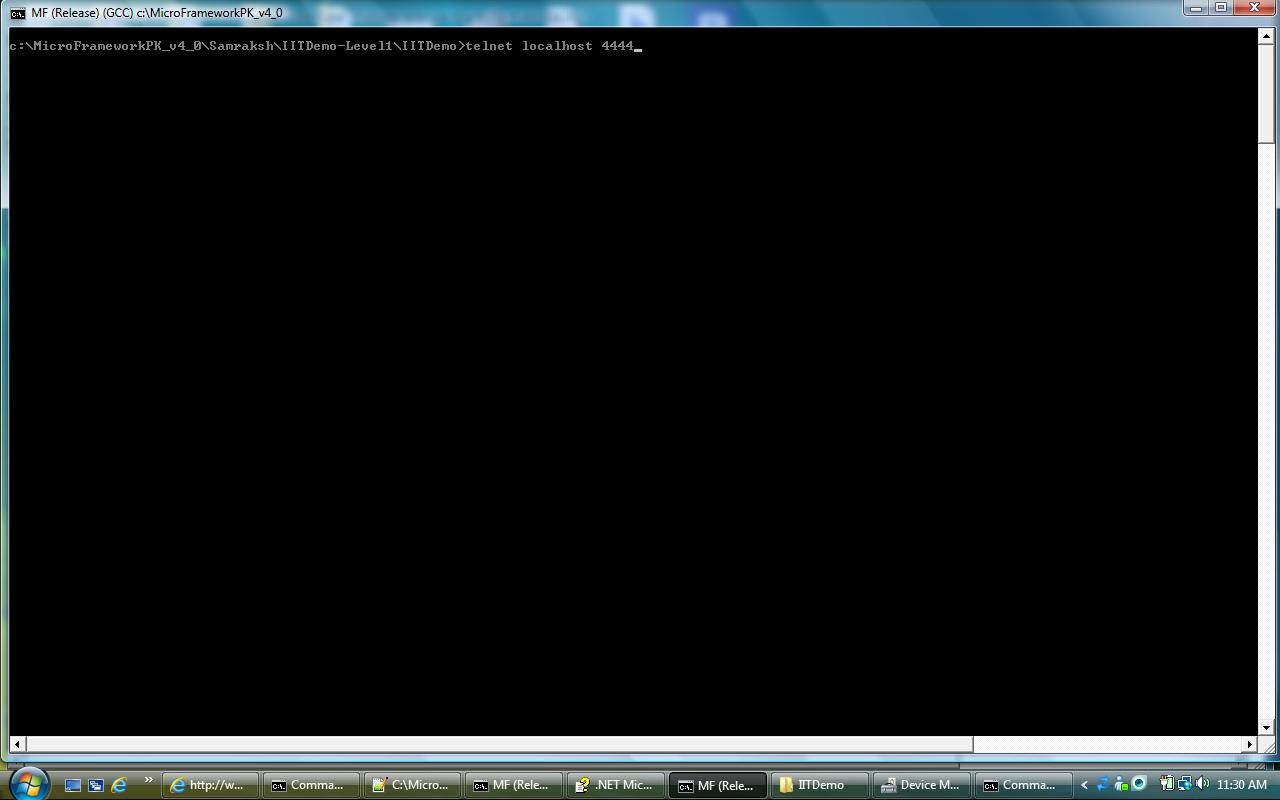
If the server starts successfully you will get the message shown below.



3. Starting the Telnet Service:

**telnet localhost 4444**

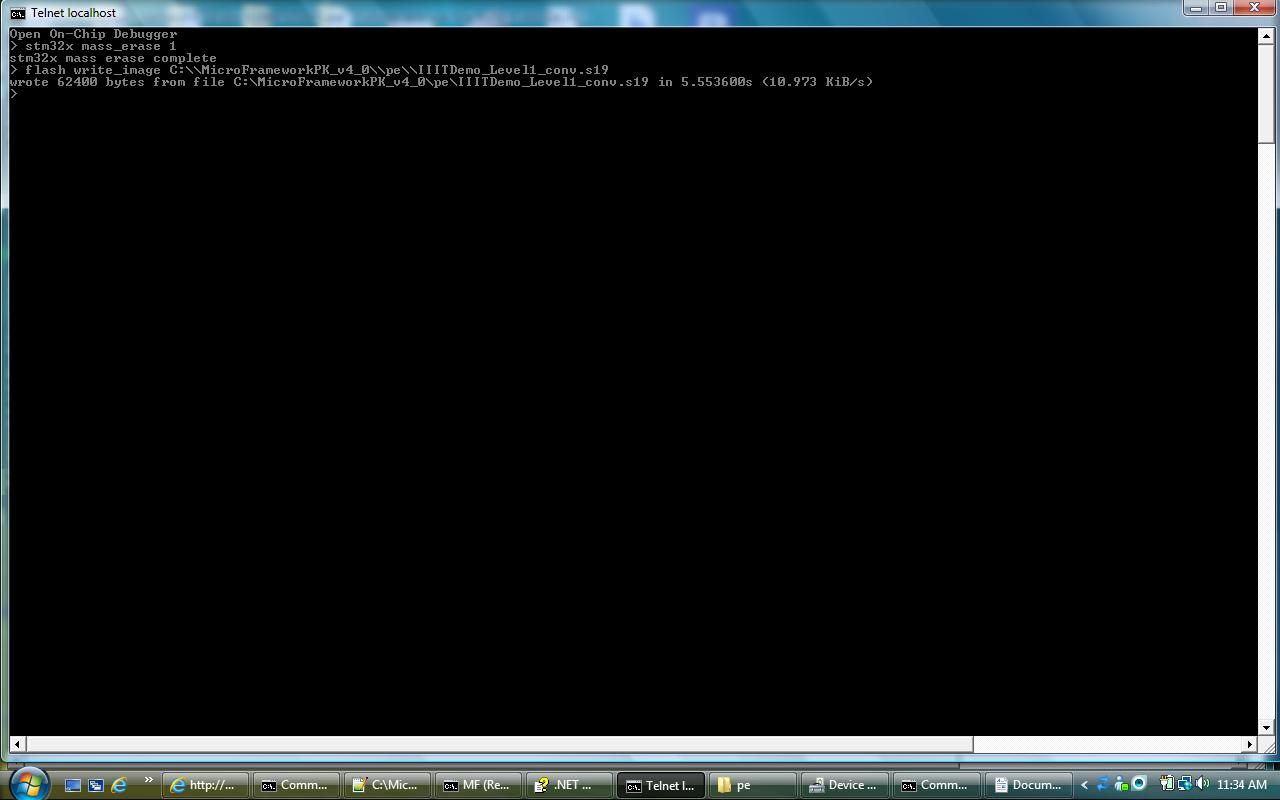
This command the telnet interface.



4. Flashing the c# app (**Note:** Do Not write to flash if you want to keep U-boot)

**flash write\_image <path>filename.s19**

This command writes the c# app to flash



5. Loading Interpreter

5a. Loading Interpreter to RAM directly. (make sure you first did a RAM build)

**soft\_reset\_halt**

resets the cpu and halts it.

**file** <path\_to\_tinyclr.bin>

**load**

**exit**

**soft\_reset\_halt**

resets the cpu and halts it.

5b. Load Interpreter to Flash **(Do Not write to Flash, if you want to kep U-boot)**

**flash write\_bank 0 <tinyclr.bin> 0**

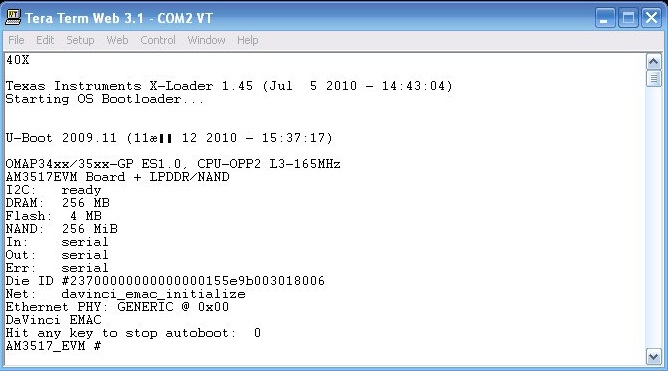
writes the interpreter to the flash, note that the command here is write\_bank and not write\_image as was the case in during the application flashing

**exit**



6. Now Power cycle the mote and the application starts running.

**3.0 Downloading and Running a Application on Soc8200 through U-boot**



1. Connect a serial cable from PC to Soc8200

2. Start a Terminal software on the COM port, like TERATERM (for Windows) and MiniCom (for Linux)

3. Configure the serial port at baud rate of 115200

4. Turn On the Soc8200, you should see output from "Texas Instruments X-Loader" on the screen

5. You will see a message "Hit any key to stop auto boot" and then a count down, hit some key to stop the autoboot.

6. Once you hit a key you will be taken to the U-boot AM3517\_EVM #prompt (as shown in figure below)

7. Build the app as a \*.srec (S-Record) file

8. Type 'loads' on the u-boot prompt, the prompt will wait for recovering a file

9 To send file,

- On Windows/Tera Term, go to file > send file> select the \*.srec file and click open

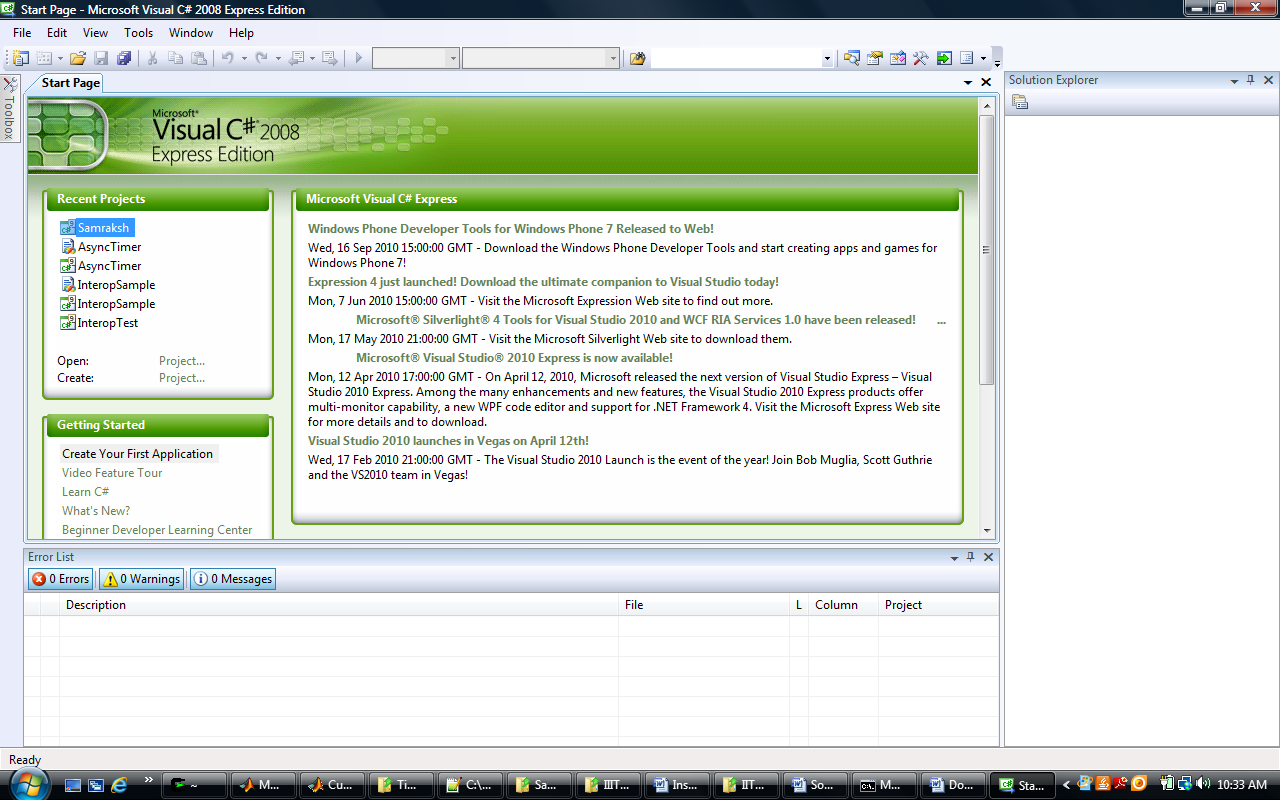
- On Linux/MiniCom, press Ctrl+A+Z, select send file and then ASIIC

10. Once file download is complete, U-boot will display a start address for your app

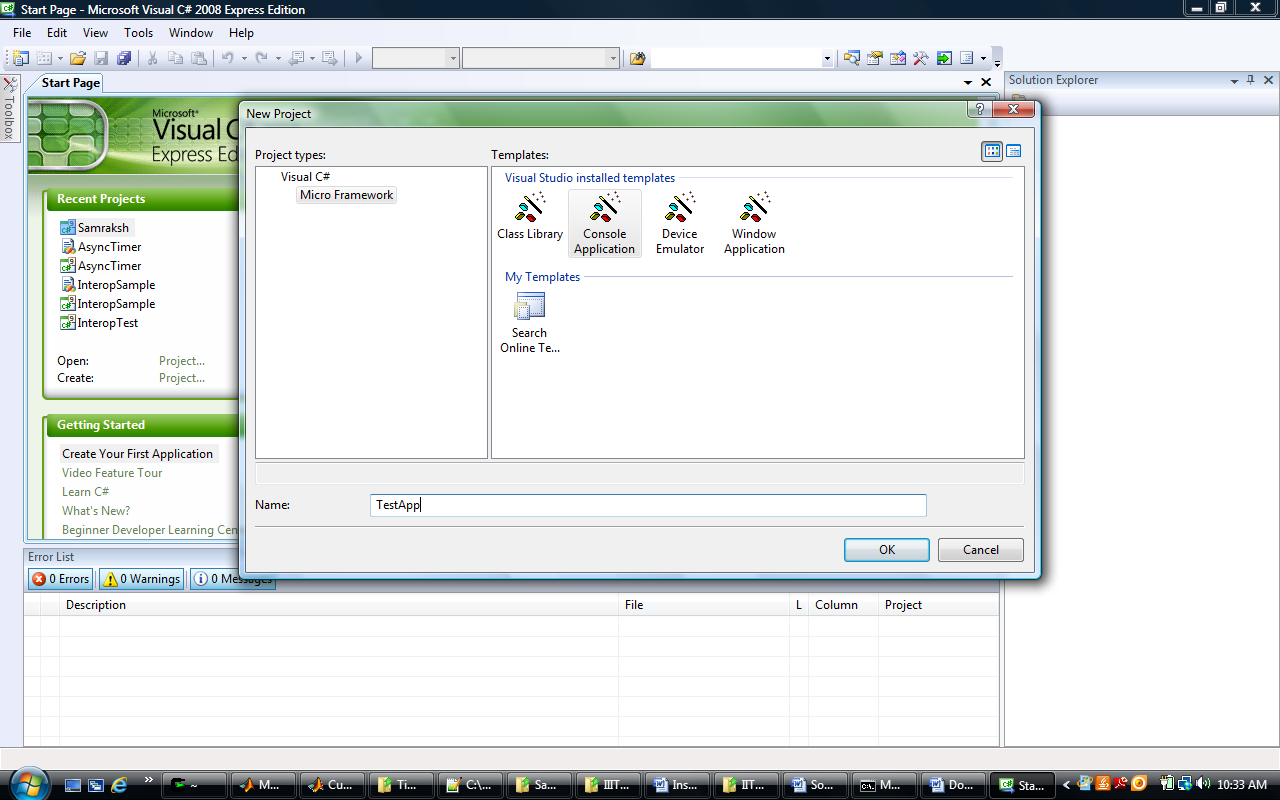
11. Type 'go start-address'

**4.0 Building C# Applications**

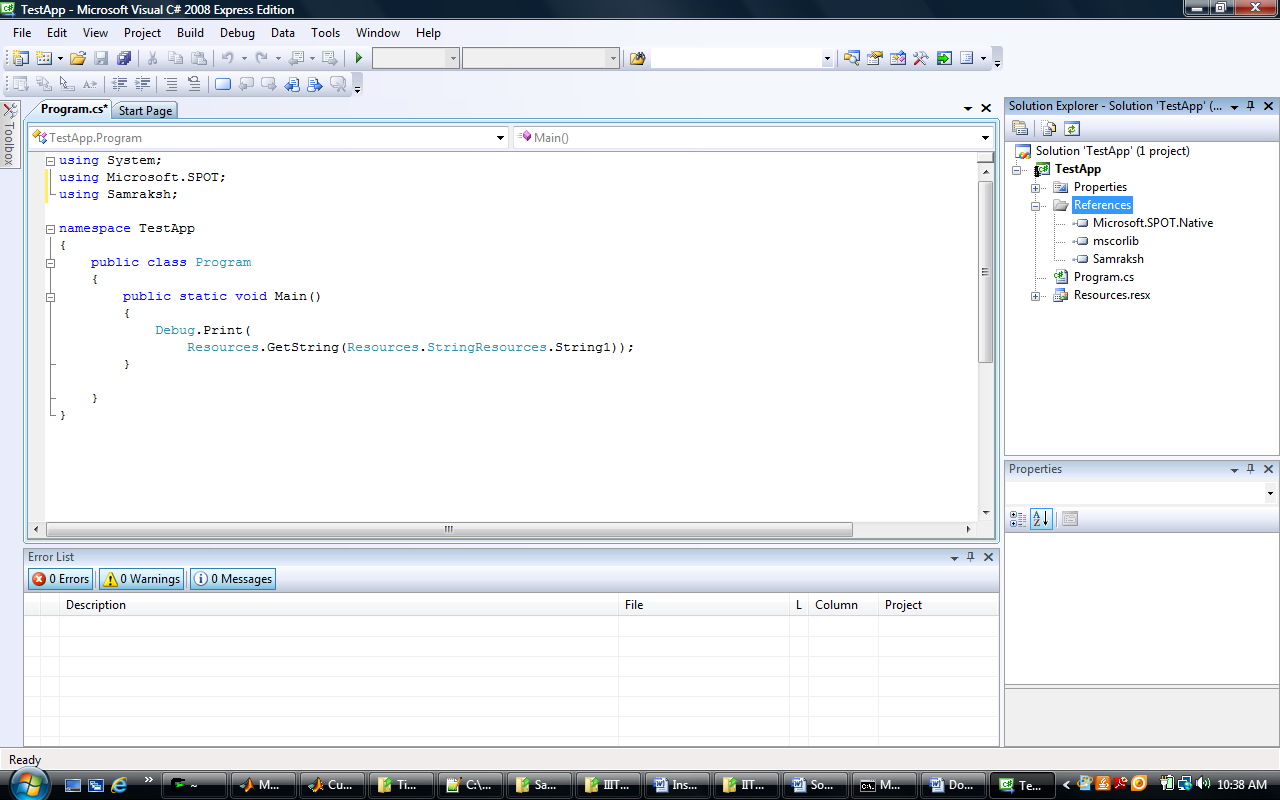
1. Create a new application in visual studio.



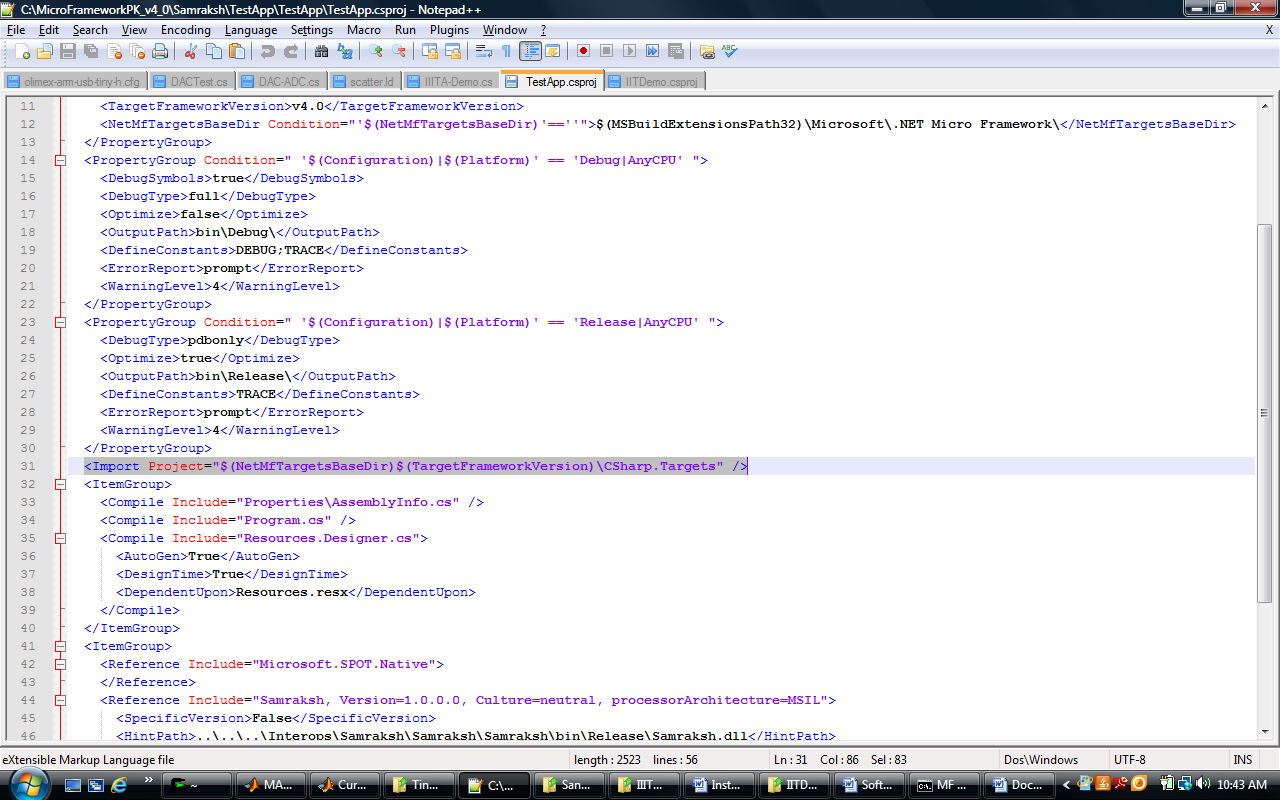
2. The application is a console application



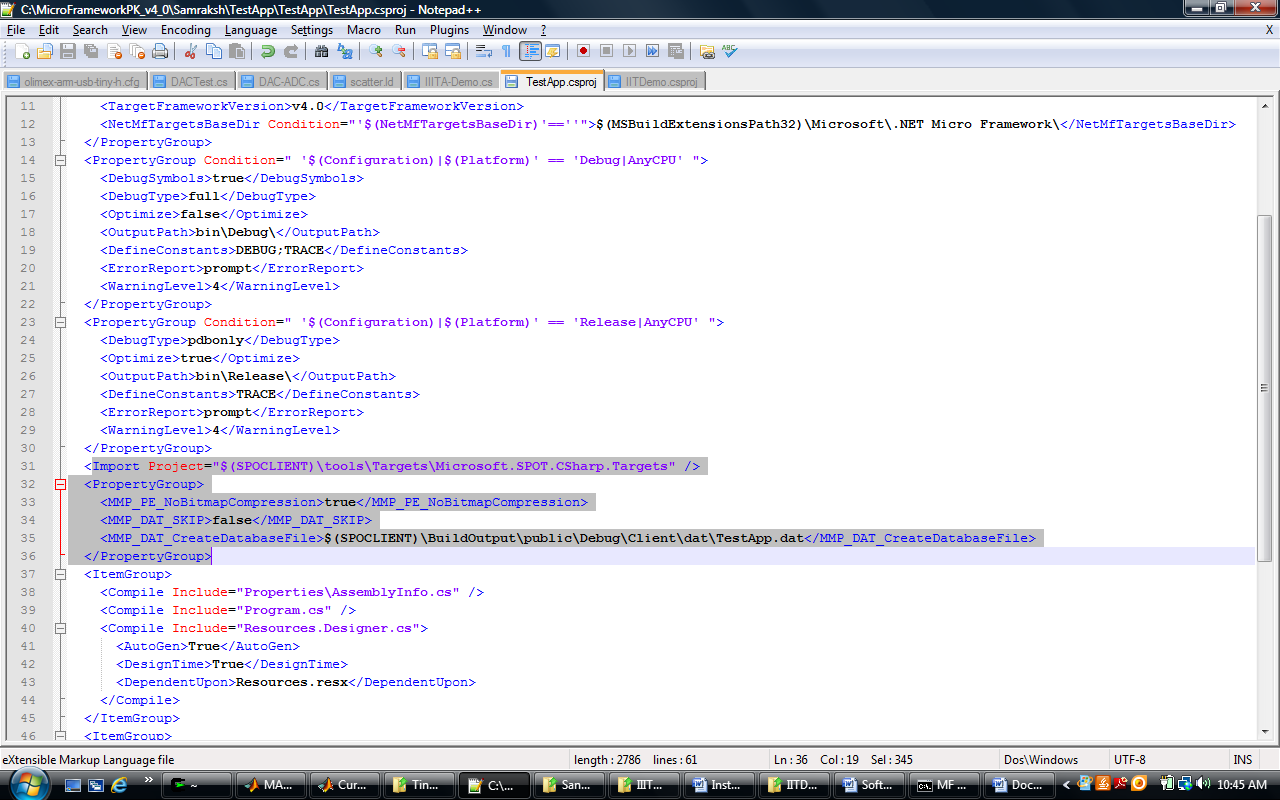
3. This generates a project as shown below.



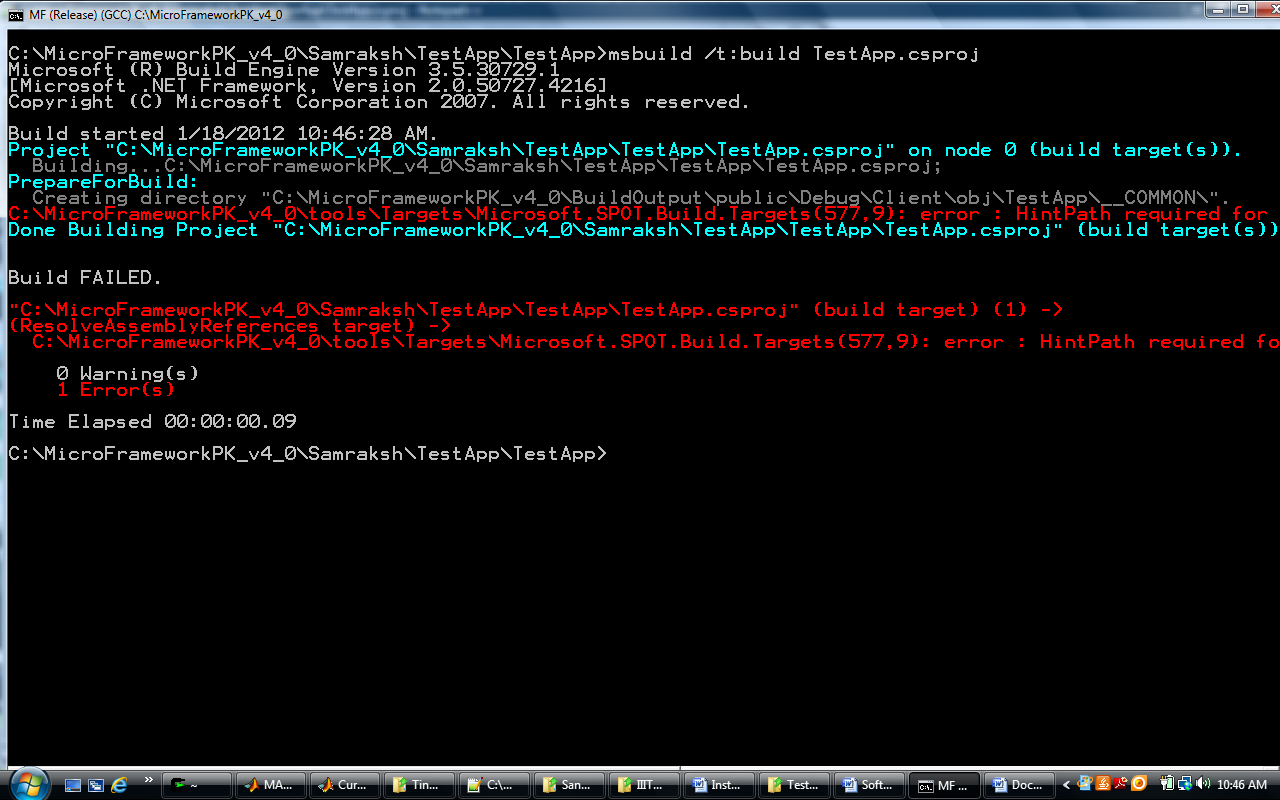
4. Once you have written the application, close visual studio and open the .csproj file corresponding to the application you have created. Replace the highlighted line with the lines shown in the next figure.



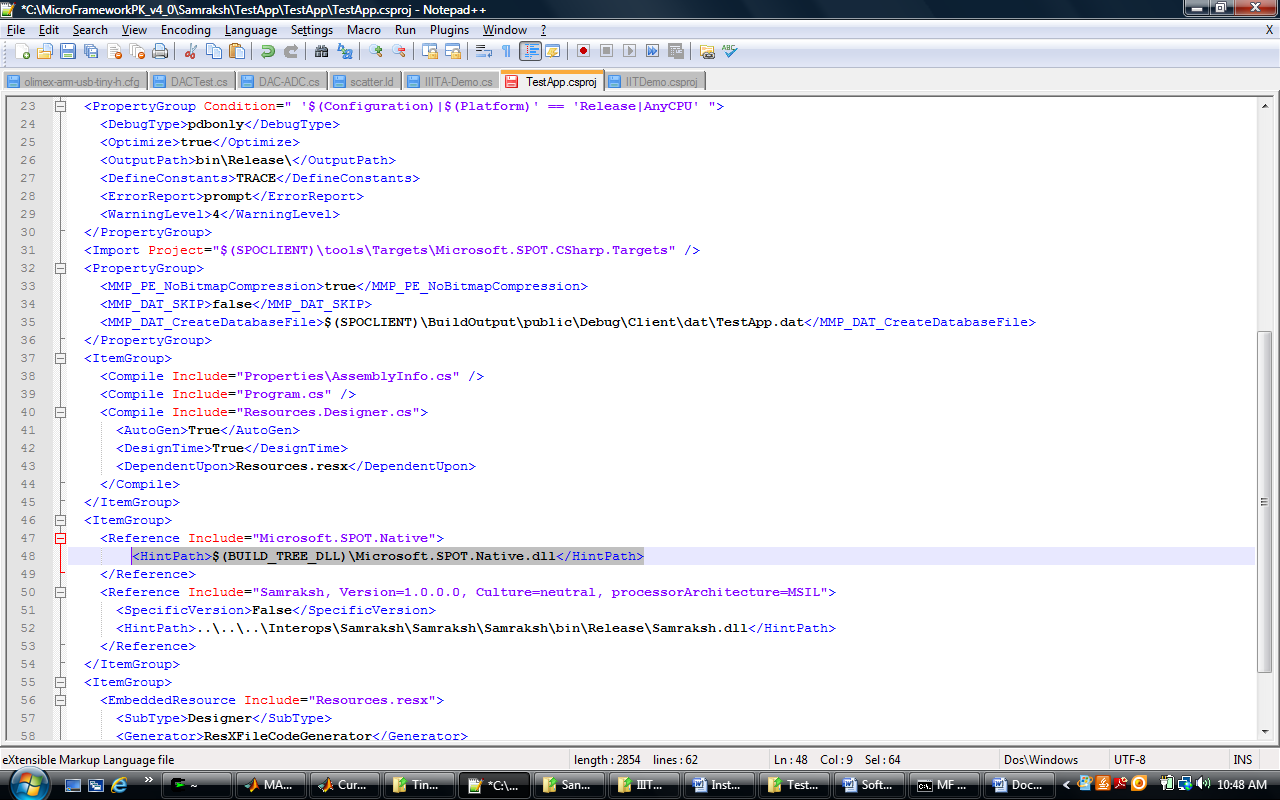
The objective here is to modify the project to create a .dat file which is used as the executable in future steps.



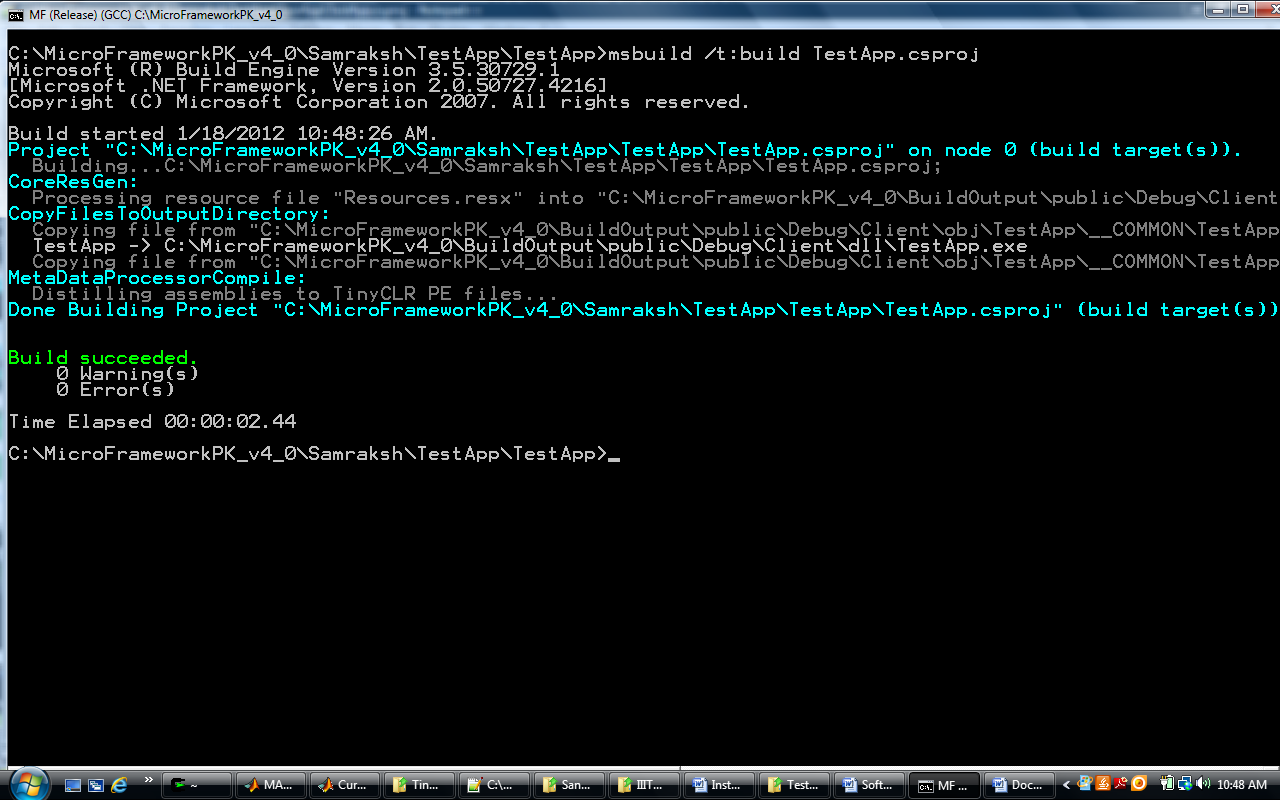
5. Now build the application from the command line using msbuild as shown below.



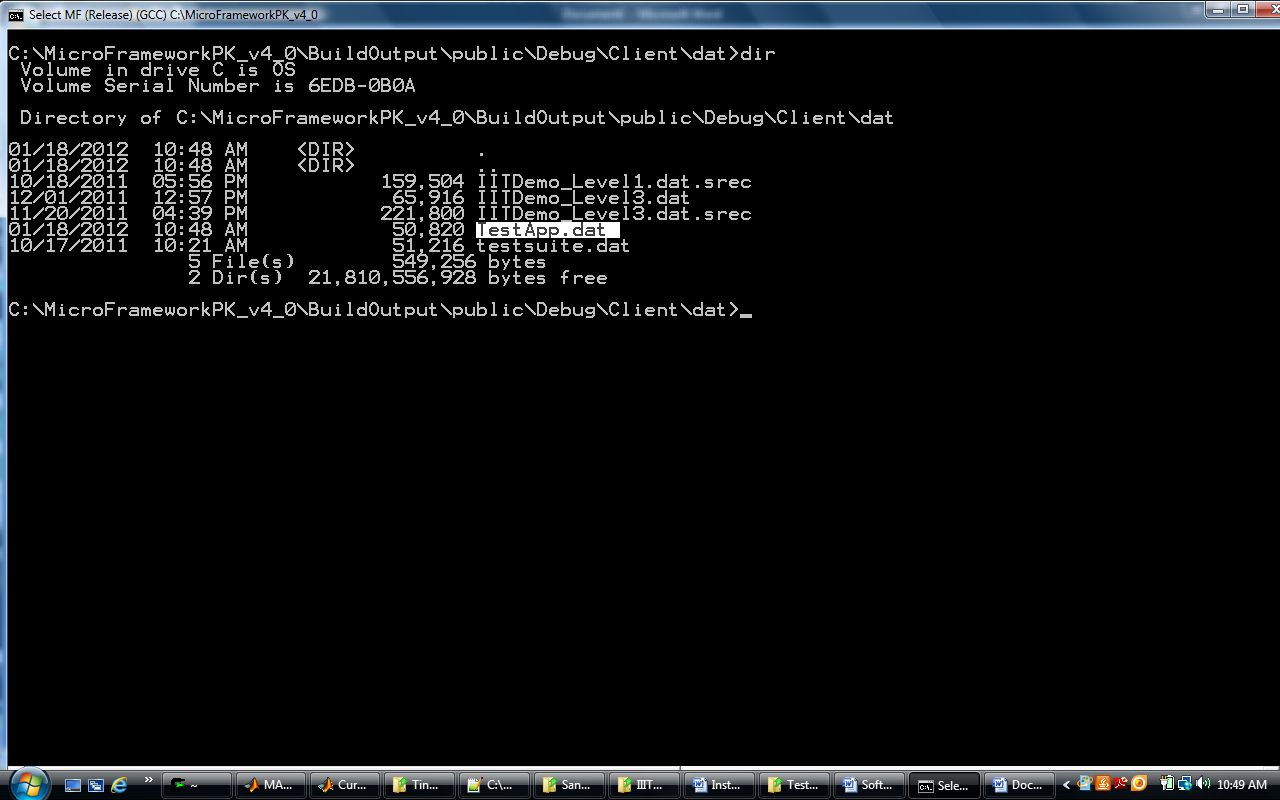
6. Build complains saying the HintPath is required. Open the .csproj file once again and add hint paths for all references including the Samraksh.dll if not already present.



7. Now build the application once again.



8. If the build succeeds without errors you will find the dat file in the folder below.



9. The next step is to convert the dat file to s19 which can be used to flash the mote. The command for this is

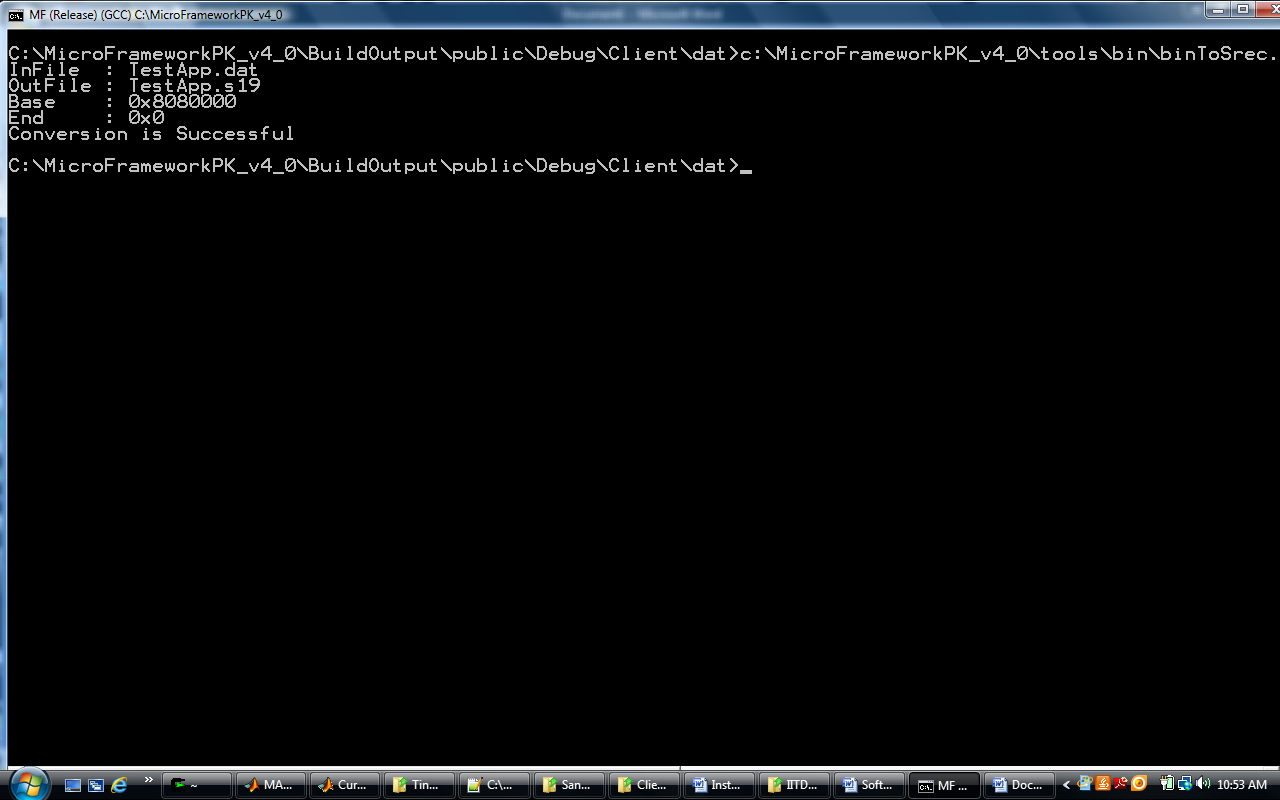
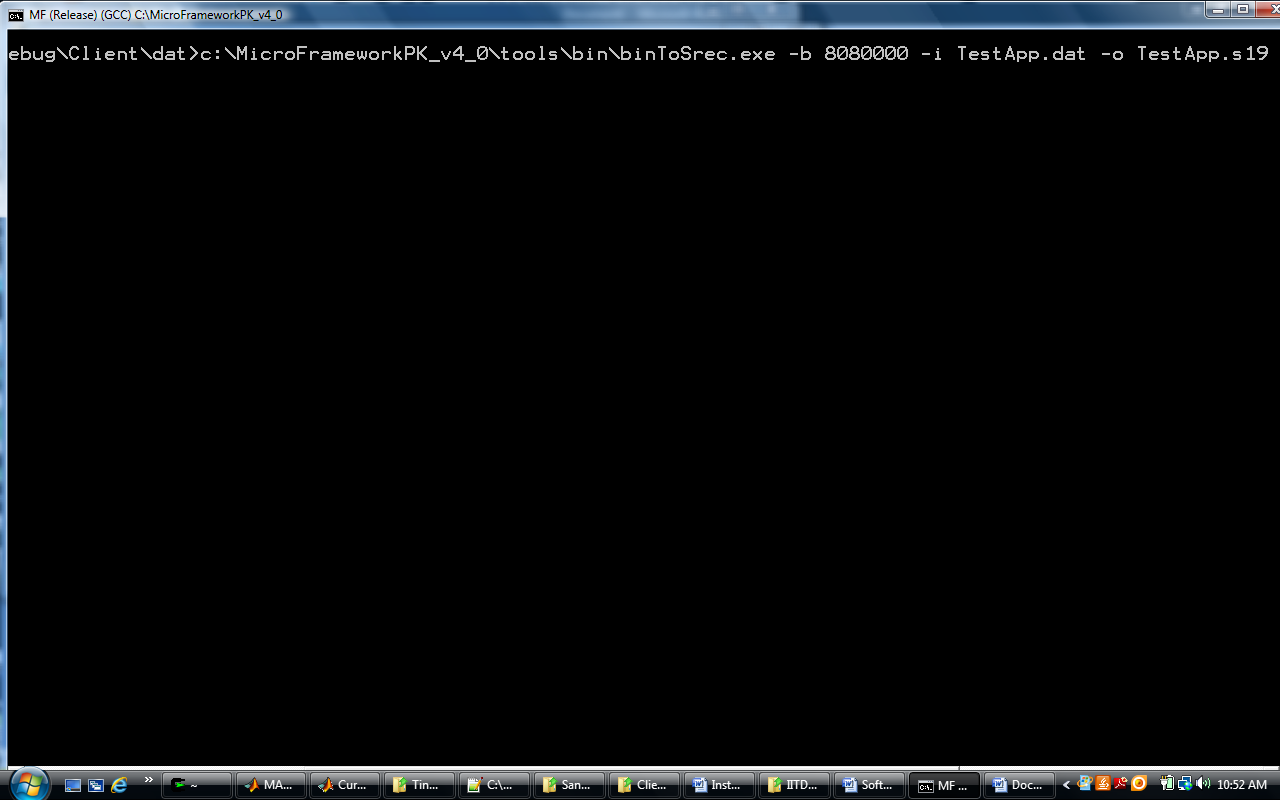
binToSrec.exe –b 8080000 –i TestApp.dat –o TestApp.s19

8080000 represents the starting address of the C# binary

TestApp.dat represents the input file and

TestApp.s19 represents the output file

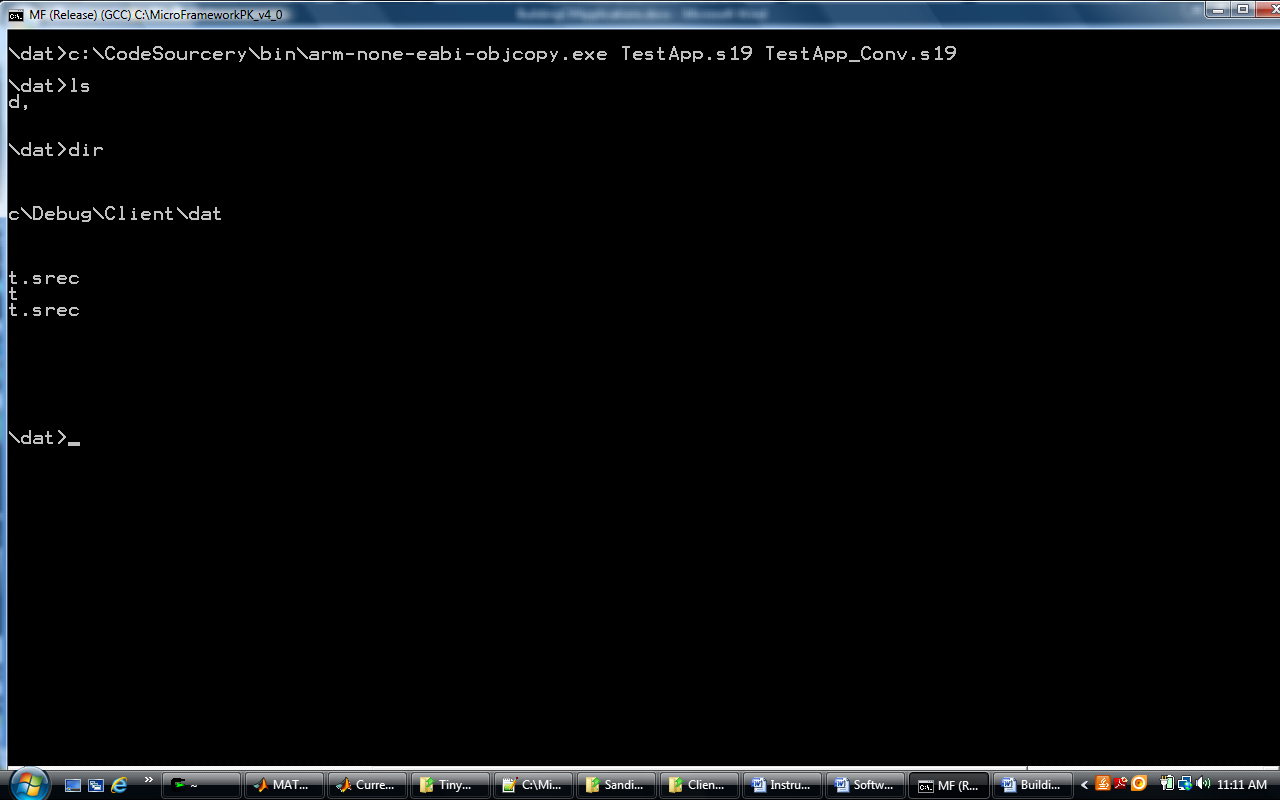
(If you do not have micro framework installed, binToSrec.exe is attached with this packet)



10. The s19 file generated by step 9 is not the right format and needs to be converted through the command below.

Arm-none-eabi-objcopy.exe TestApp.s19 TestApp\_Conv.s19

(If you do not have codesourcery lite installed, you can either obtain it online or I am attaching arm-none-eabi-objcopy.exe with this packet)



11. The result of this step can now be loaded on to the mote. The instructions on how this is done is available in the document SoftwareDownloadToNode.docx.