

SETUP MANUAL FOR THE .NET MICRO FRAMEWORK ON THE RENESAS SH7619 EVB

Release Notes, version 1.2

Revision #	Revision Date	Notes
1.0	June 2, 2010	Initial document
1.1	June 10, 2010	Reviewed and updated flashing instructions
1.2	Oct 4, 2010	Updated for online release

OVERVIEW

SCOPE

This document describes how to use the .NET Micro Framework (.NETMF) with the SH7619 EVB board. This document mainly describes how to setup, build and run the SH7619_EVB solution available in the porting kit. This document also describes how to create, build, deploy and run the .NETMF based applications on the SH7619 EVB board using Microsoft Visual Studio.

REQUIRED TARGET SYSTEM

- SH7619 EVB

Please read through this document before powering ON and working with the board.

SUPPORTED DRIVERS

- Serial
- On chip Ethernet
- Timer & Power
- NOR Flash

The following drivers are just samples. In the SH7619 EVB there is no LCD panel and no Key Inputs. If you do some of your own wiring to the board, the following sample drivers might help you to create a driver.

- Display (LCD : 132 x 176 TFT LCD Display (HD66773R) with 262,144 colors)
- Key input using GPIO.

SUPPORTED PROJECTS

Following projects are supported.

- NativeSample
- TinyCLR

Following projects are not supported.

- Portbooter
- Tinybooter

However, these projects can be built properly, so if needed you can customize them.

REQUIREMENTS

- Host system (Windows XP in this documentation)
- Target System (SH7619 EVB)
- Microsoft Visual Studio 2010
- E10A-USB Emulator for SH2 and installation setup.
- C/C++ Compiler Package for SuperH RISC engine family
- .NET Micro framework porting kit (MicroFrameworkPK.msi)
- .NET Micro framework development kit (MicroFrameworkSDK.MSI)
- .NETMF Solution “SH7619_EVB” (included in the porting kit)
- NULL modem cable (Serial cable)
- Tera Term Pro or another terminal emulator program.

SETUP

PREPARATION

Before setting up the Porting Kit, the software below has to be installed.

- Visual Studio 2010
- Renesas C/C++ Compiler Package for SuperH RISC engine family V.9.02 Release 00 or later

For these instructions, we will assume it is installed in

`c:\Program Files\Renesas\Hew\Tools\Renesas\Sh\x_y_z*`

INSTALL THE .NET MICRO FRAMEWORK PORTING KIT

The .NET Micro Framework Porting 4.1 Kit is available as a download from the Microsoft® Download Center. See the link: <http://www.microsoft.com/downloads/en/details.aspx?FamilyID=ccdd5eac-04b1-4ecb-bad9-3ac78fb0452b>

Install the porting Kit by executing MicroFrameworkPK.msi. For these instructions, we will assume that this is installed in:

C:\MicroFrameworkPK_v4_1

In addition, one could also install the TCP/IP and SSL Libraries for the SH2/A Instruction Set and/or the Cryptographic libraries, also available from the Microsoft® Download Center.

Necessary updates to the porting kit for SH7619 support are available at the TrygTech website:

<http://www.trygtech.com/downloads/sh7619.php>

Download "NETMF41_SH7619_Updates.zip" and extract, overwriting existing files, to c:\MicroFrameworkPK_v4_1.

SET PATH

- 1) Open a command-line prompt
- 2) Change the current directory to
C:\MicroFrameworkPK_v4_1
- 3) Enter the line below,
setenv_shc "c:\Program Files\Renesas\Hew\Tools\Renesas\Sh\x_y_z*"

*x_y_z is the version of the compiler being used; for example 9_2_0.

Memory Map

The memory map of the SH7619 EVB .NETMF Porting kit is shown below.

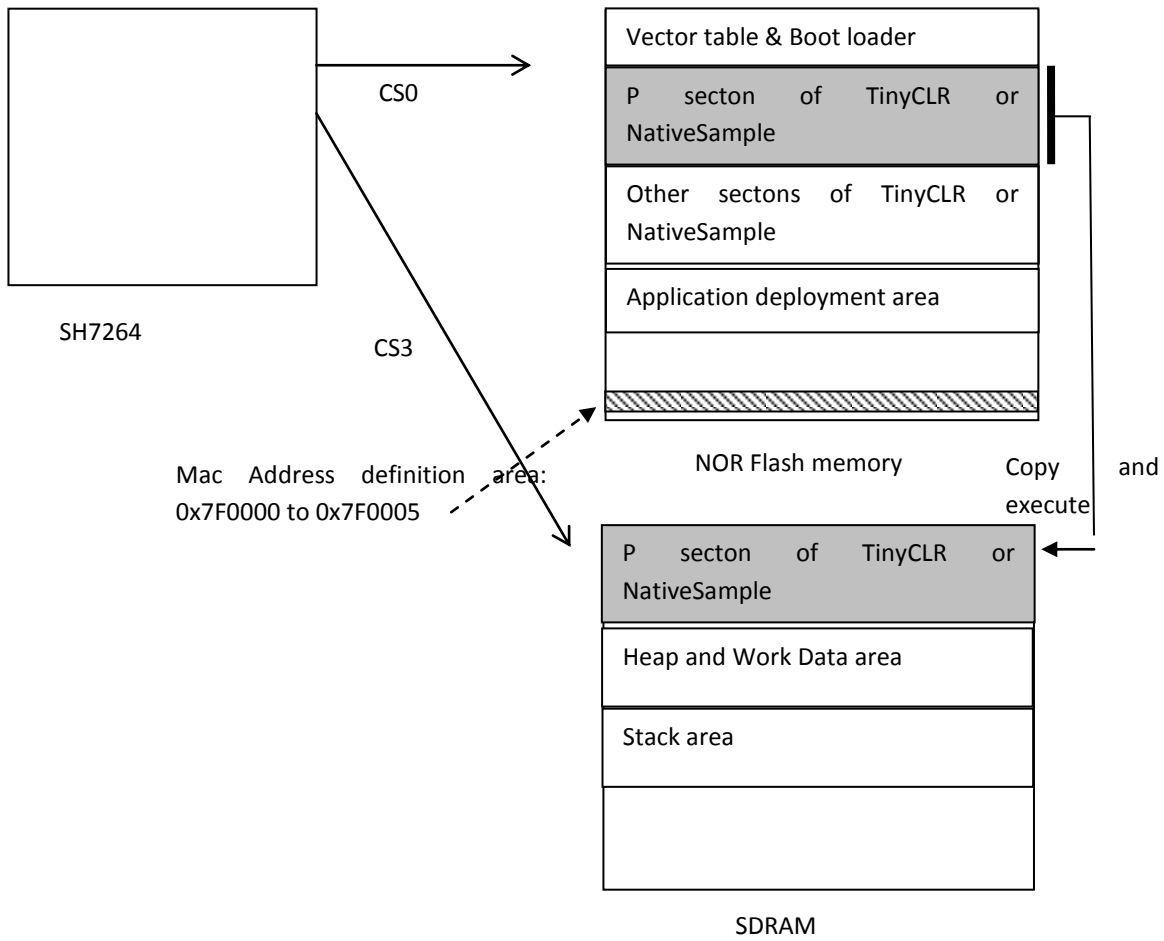


FIGURE 3.1 MEMORY MAP OF THE SH7619 EVB .NETMF PORTING KIT

How to build and execute

In this section, we will describe the way to build, download and execute the SH7619 EVB solution available in the porting kit with NOR Flash Memory and SDRAM.

HOW TO BUILD

1) Using the Command-line prompt, change the directory to “Solutions\SH7619_EVB”

```
C:\MicroFrameworkPK_v4_1>cd solutions\SH7619_EVB
```

2) Run

```
Msbuild dotnetmf.proj /t:build /p:flavor=debug;EnableTcpIp=true
```

Flavor: <debug|release|rtm>

EnableTCPIP : <ture|false>

If you want to debug your program using the E10A-USB Emulator, please specify “debug” for the “flavor” option.

BOARD SWITCH SETTINGS

Set SW1 of the SH7619 EVB board as below to startup from NOR Flash.

SW	Setting	Function
1	ON (Low)	MD0
2	OFF (High)	MD1
3	ON (Low)	MD2
4	ON (Low)	MD3
5	ON (Low)	MD5

The meaning of above setting is below,

- Clock mode
Clock Mode is MODE2.
Each frequency is below,
 - Crystal 15.36MHz
 - CPU 15.36MHz
FRQCR register is set by software as 0x1103 so the CPU frequency will be 122.88MHz.
 - BUS 61.44MHz
 - Peripheral 30.72MHz

- CS0 Memory bus width
16 Bit

- Endian
Big endian

DOWNLOAD USING E10A-USB

1. Setup the E10A-USB Emulator

Install the E10A-USB Emulator software into your PC. During the install, you should select the device group for the E10A-USB then specify "Super H RISC engine family SH-2 device group"

2. Make sure that the switches (beneath the sliding panel) on the top of the E10A-USB are set to the correct positions. Switches 1 and 2 must be in the "1" position and switch 3 must be in the "0" position.

3. Start downloading the .NETMF to the SH7619_EVB using the E10A-USB and the instructions below:

Extract the HEW workspace for SH7619 EVB Porting Kit from SH7619board.zip to any place in your PC.

In the following instructions, we will assume it is installed in "c:\workspace"

Start the Hew by choosing the menu below

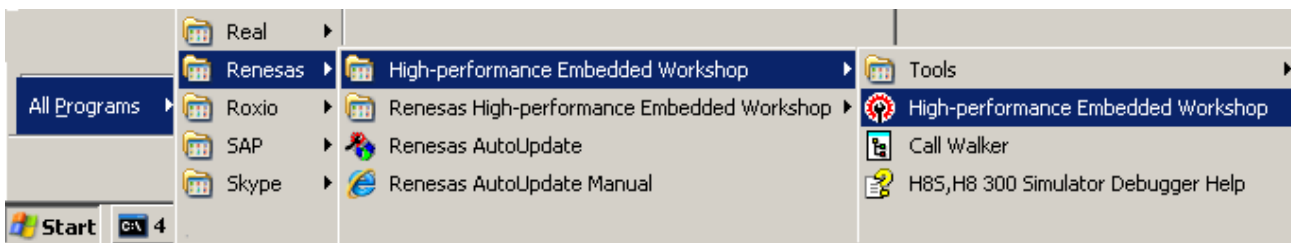


FIGURE 4.1 HEW MENUS

The Hew will then come up and you will see the dialog box shown below.

Specify the workspace as shown, and then press the OK button.

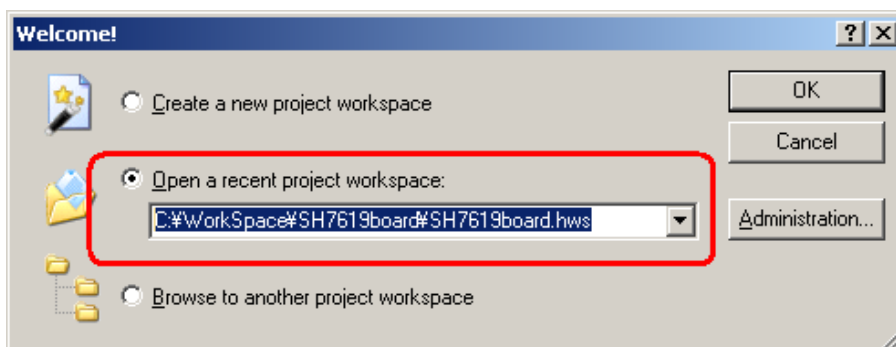


FIGURE 4.2 THE WELCOME DIALOG BOX

In the dialog box shown below, choose "SH7619".

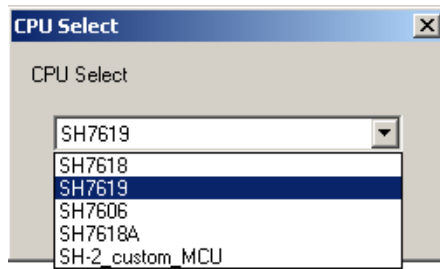


FIGURE 4.3 CPU SELECT DIALOG BOX

When the message shown below is displayed, power ON the target board and then press the OK button.

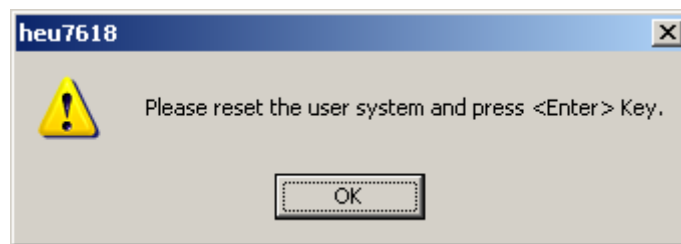


FIGURE 4.4 HEU7618 DIALOG BOX

The Hew/E10A-USB will then finish connecting to the SH7619.

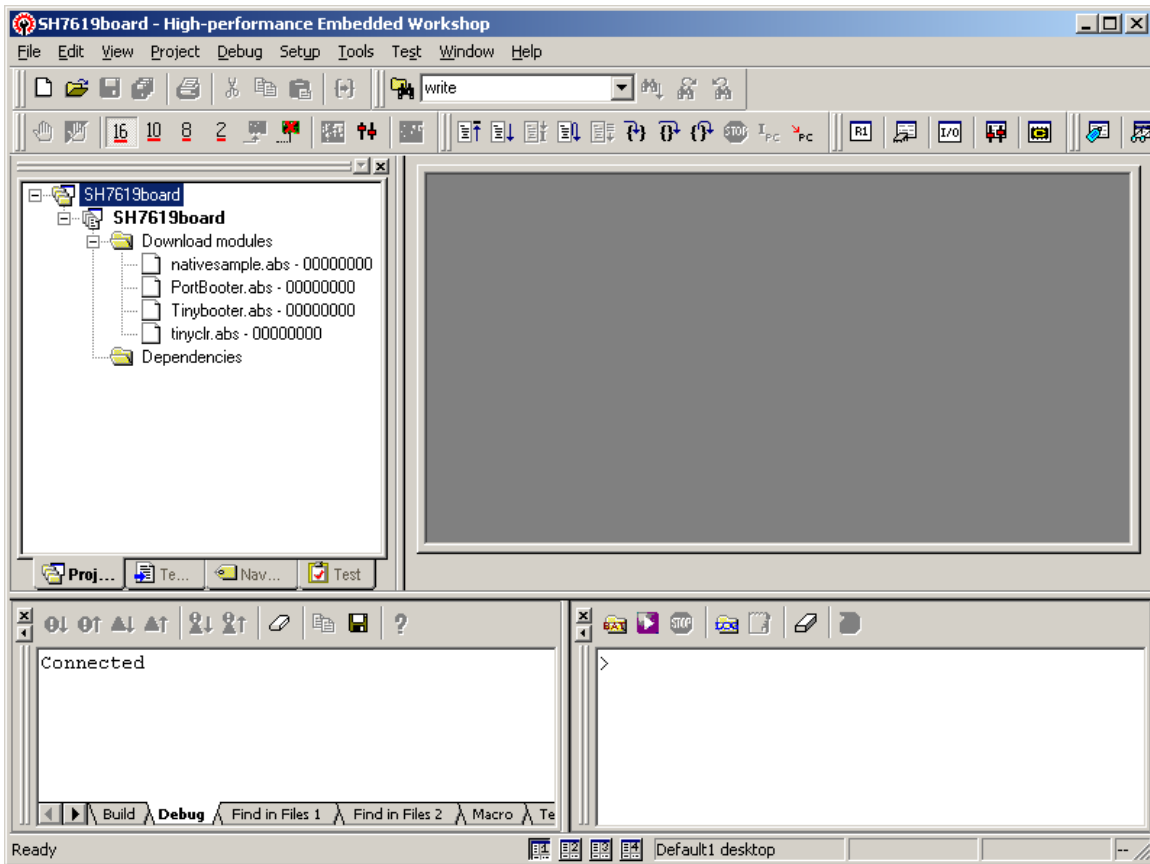


FIGURE 4.5 THE HEW WINDOW

3) How to download the program to flash memory

i) Prepare the download module.

Select [Debug] -> [Debug Settings...] from the Hew menu bar. The dialog box shown below will then be displayed.

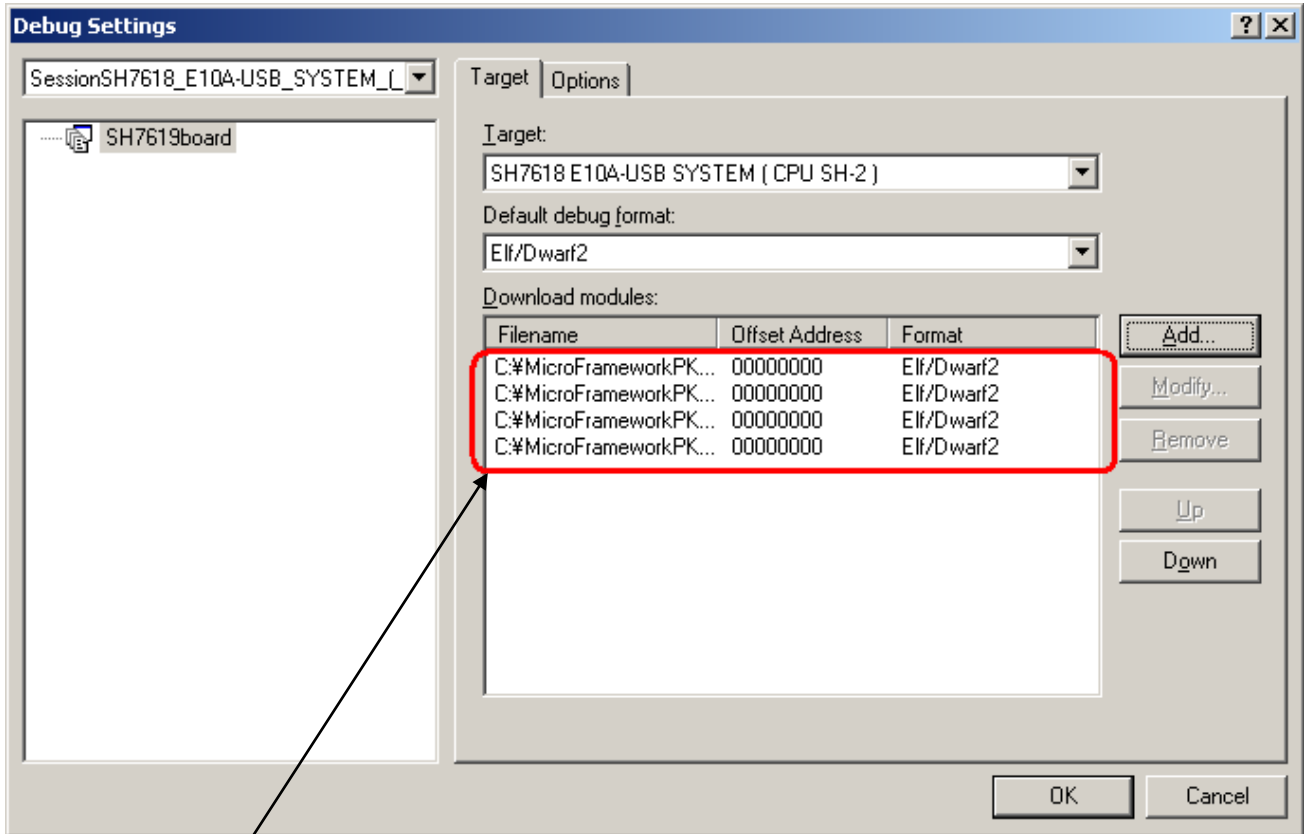


FIGURE 4.6 DEBUG SETTINGS DIALOG BOX

Here is the list of download modules. Please change the Path setting for each download module by clicking the "Modify" button.

ii) Prepare for download to Flash Memory.

Select [Setup] -> [Emulator] -> [System...] from the Hew menu bar. The "Configuration" dialog box will be displayed.

Select the "Loading flash memory" tab, and then change the dialog entries as shown in the figure below.

For "File name", please specify

<Workspace folder>\SH7619board\Tools\4MB\fmtool.mot.

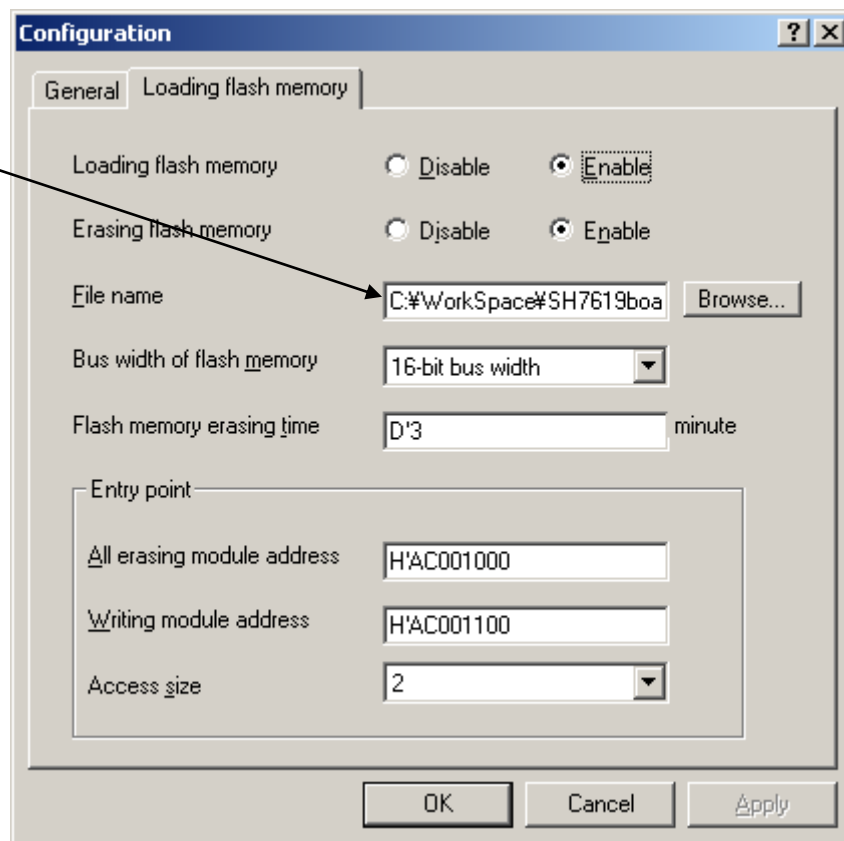


FIGURE 4.7 CONFIGURATION DIALOG BOX

NOTE that this step must be performed each time prior to a download. This emulator configuration setting is not saved. If you do not enable "Loading flash memory" prior to downloading a pattern to the board, the download will appear to take place, however the pattern on the board will remain unchanged.

ii) CPU and memory initialization

In order to download data to flash properly, CPU and memory should be initialized as following.

- CPU : Cache must be disabled.
- Memory : SDRAM must be initialized.

In order to initialize these, run the batch file as shown below,

- Specify the batch file

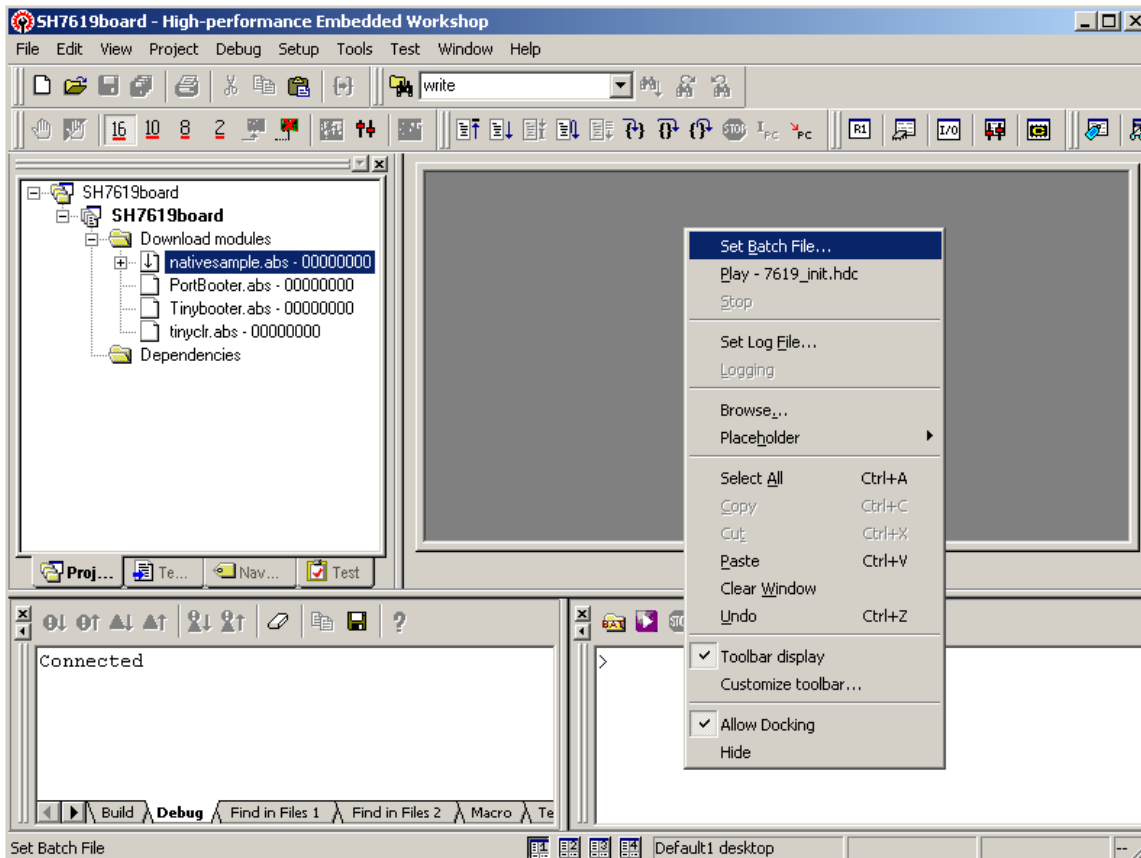


FIGURE 4.8 SET THE BATCH FILE

Perform a right-click in the command line window to display the popup menu.

Select “Set Batch file...” and then specify the batch file name as shown below,

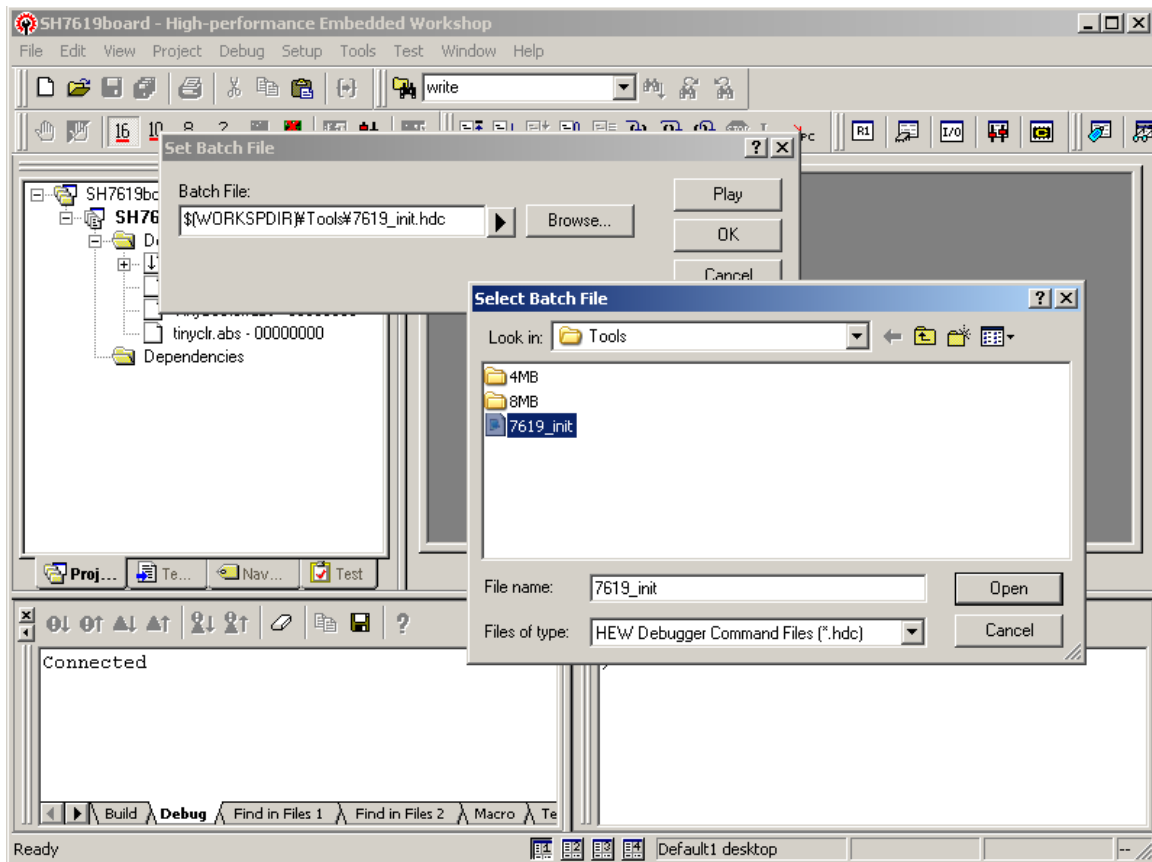


FIGURE 4.9 SET BATCH FILE

-Run the batch file

After specify the batch file, please press “Play” button so that batch file starts to run.

NOTE that powering down the board or running software on the board will require that this step be performed again.

iii) Download module to Flash

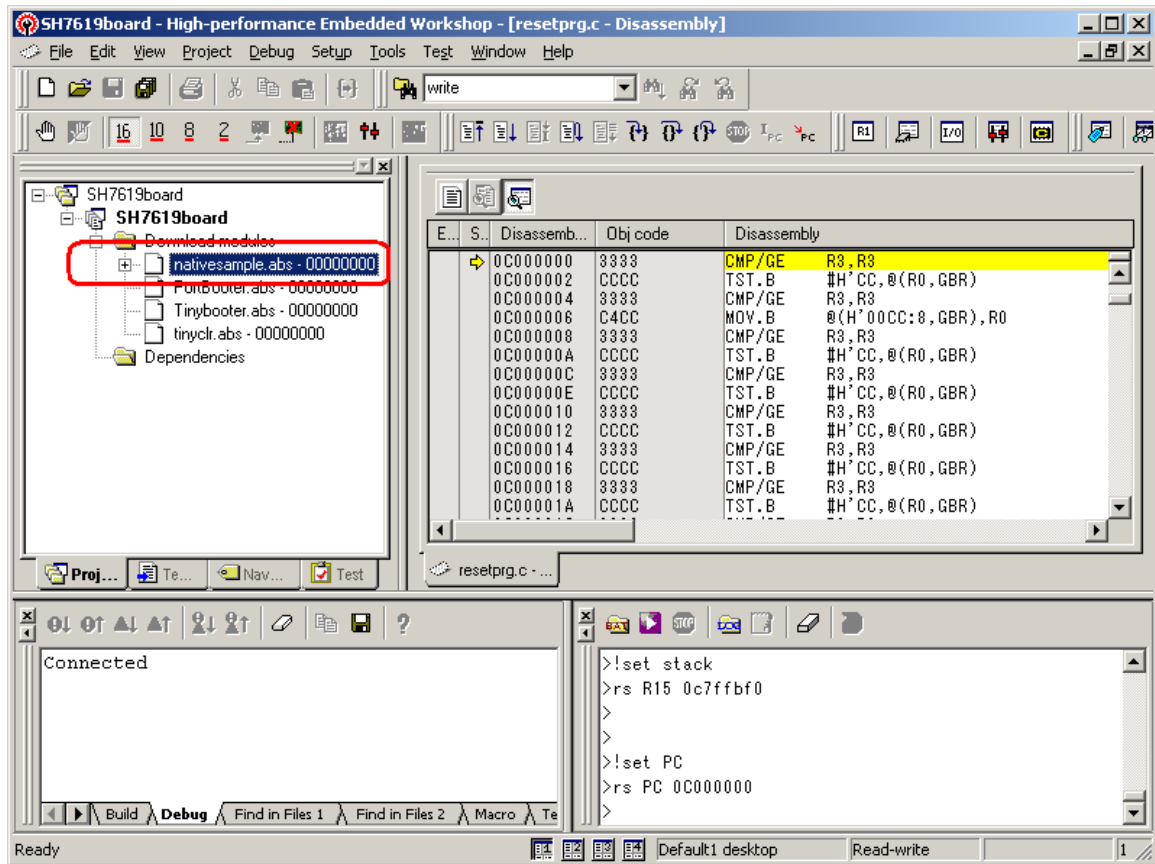


FIGURE 4.10 HEW WINDOW

In order to download, double click on the name of the download module which you want to download to flash. It takes more than 30 seconds.

In the case of downloading TinyCLR, please select tinyclr.abs instead of nativesample.abs.

RUNNING TINYCLR

Hit the “Stop” button in the toolbar or Select the “Halt Program” option from Debug menu. Disconnect the E10A-USB from the HEW, power OFF the board and disconnect the E10A connection from board.

To verify that TinyCLR is up and running, connect the serial port of the target with the PC and start the terminal emulator program (Tera Term Pro) with following settings:

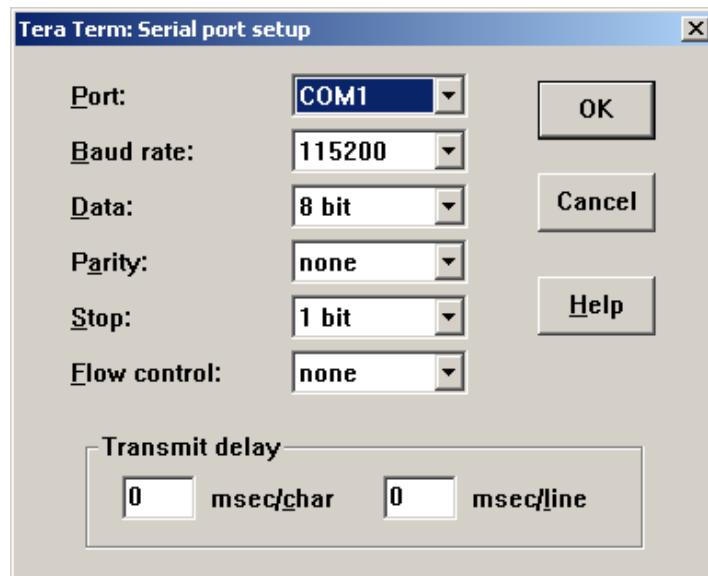


FIGURE 5.1 SERIAL PORT SETUP

Power-ON the board. You should see the text similar to the following in the terminal emulator window:

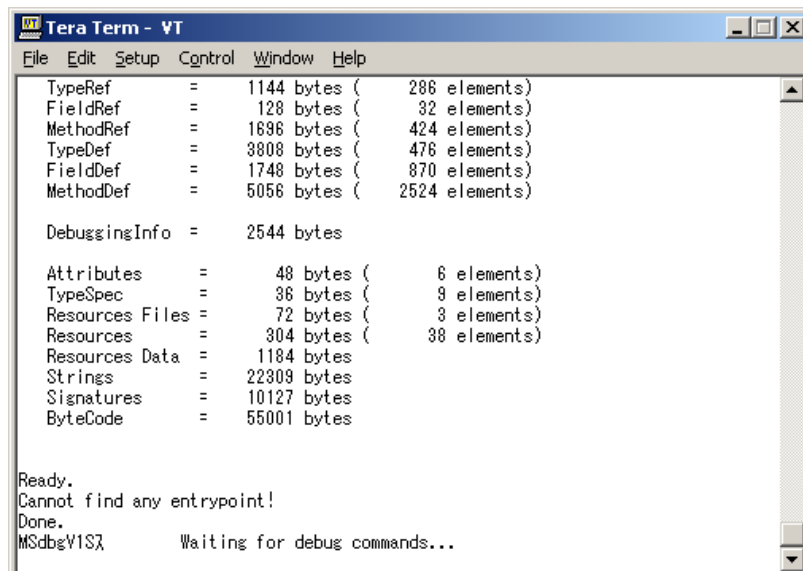


FIGURE 5.2 SERIAL TERMINAL

If what you see in the terminal emulator window looks similar to that shown in Figure 5.2 above, congratulations! Your TinyCLR is up and running. The example above uses a debug build of TinyCLR.abs.

Please note that you must close your terminal emulator before proceeding. Visual studio will be unable to communicate with your board while the terminal emulator has control of the serial port

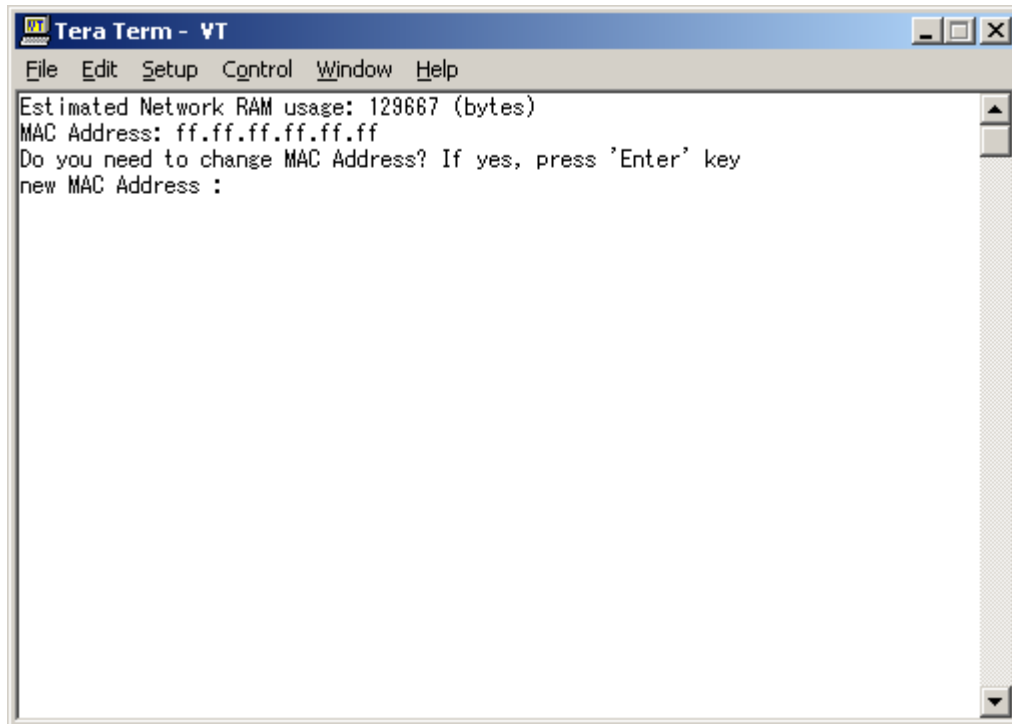
HOW TO SET THE MAC ADDRESS

The initial value of MAC address is FF.FF.FF.FF.FF.FF.

Using a terminal emulator (for instance Tera Term Pro), you can change the MAC address.

- 1) Open the terminal emulator and connect the serial port of the SH7619 EVB with PC via null modem cable.
- 2) Press the Enter key
- 3) Power on the SH7619 EVB while holding down the Enter key.

Please don't release the Enter key until you see the following message in the terminal emulator window.



- 4) Enter new MAC address.

If the expected MAC address is 1a.e.f.f8.6.f3, type the following:

1a.0e.0f.f8.06.f3

HOW TO DEPLOY AN APPLICATION

A feature of the .NET Micro Framework is the ability to develop applications using Visual Studio 2010.

This section will describe how to deploy an application by using one of the Sample applications supplied in the .NET Micro Framework porting kit.

INSTALL THE .NET MICRO FRAMEWORK SDK

In order to use the .NET Micro Framework with Visual Studio, you need to install the .NET Micro Framework SDK (MicroFrameworkSDK.msi) in your PC.

PREPARE THE SAMPLE APPLICATION

First, you need to create the application. Please open Visual Studio.

i) Create a new project

Select File menu -> New -> Project.

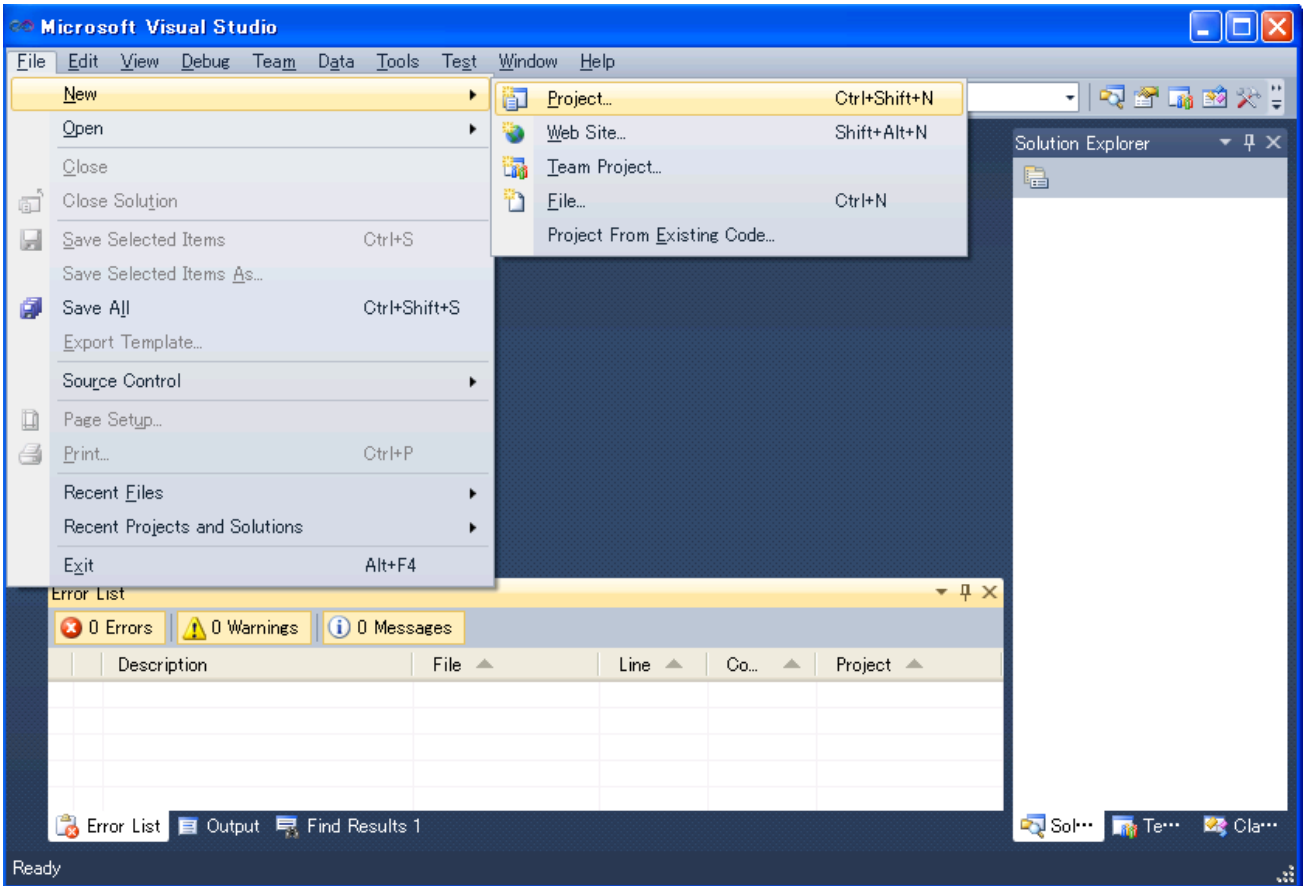


FIGURE 7.1 MICROSOFT VISUAL STUDIO

ii) Select the Project type, Template and project name

Select below,

Project Type: Micro Framework

Template: Console Application

Name: HelloWorld

And press the OK button.

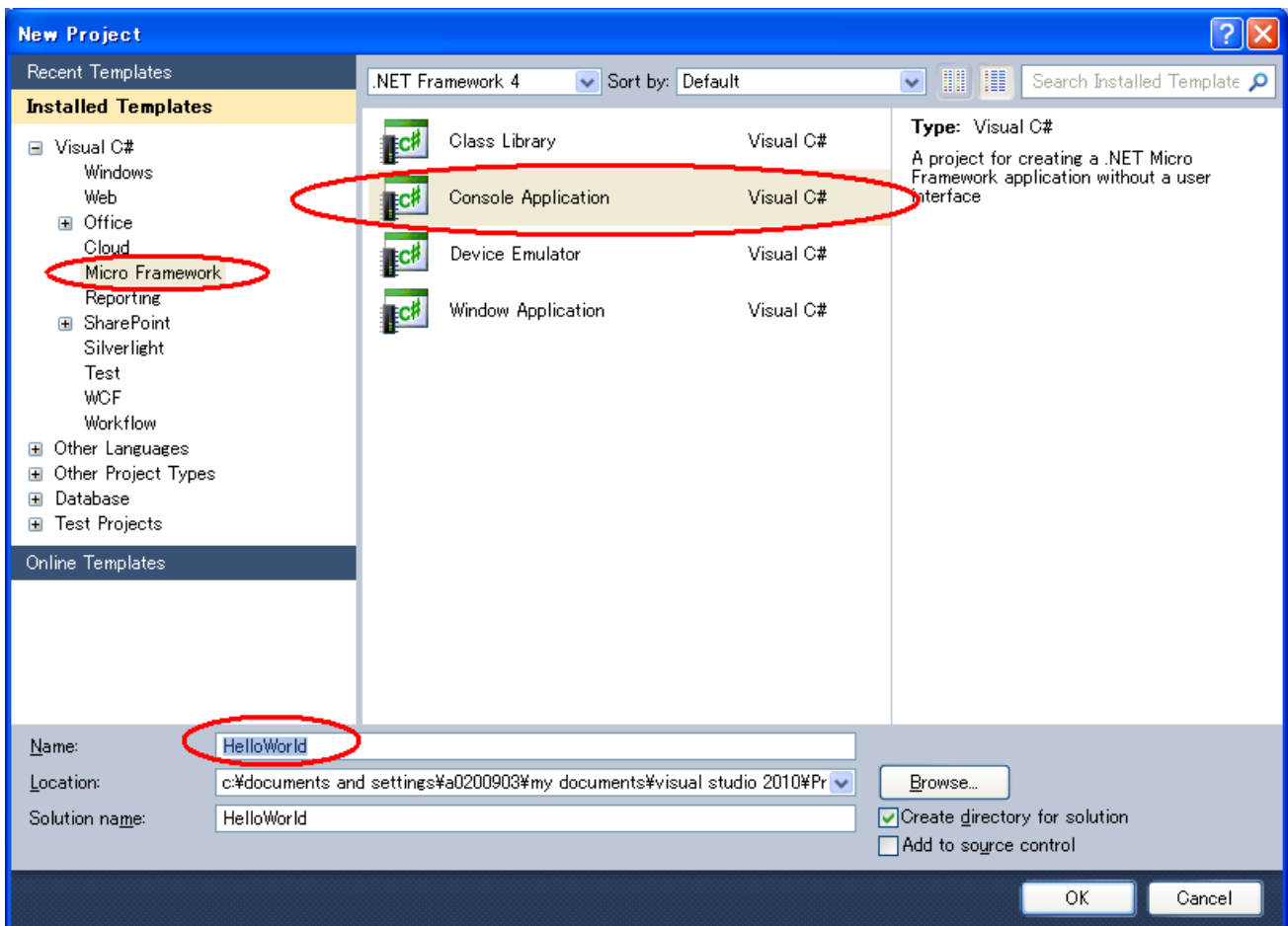


FIGURE 7.2 NEW PROJECT

Note that depending on your application, you should select the appropriate options. If you are creating a GUI based application it is advised that you select the “Window Application” option.

iii) Copy the source file for the Sample program into this project

Right click on "HelloWorld" in the Solution Explorer and select Add -> Existing Item....

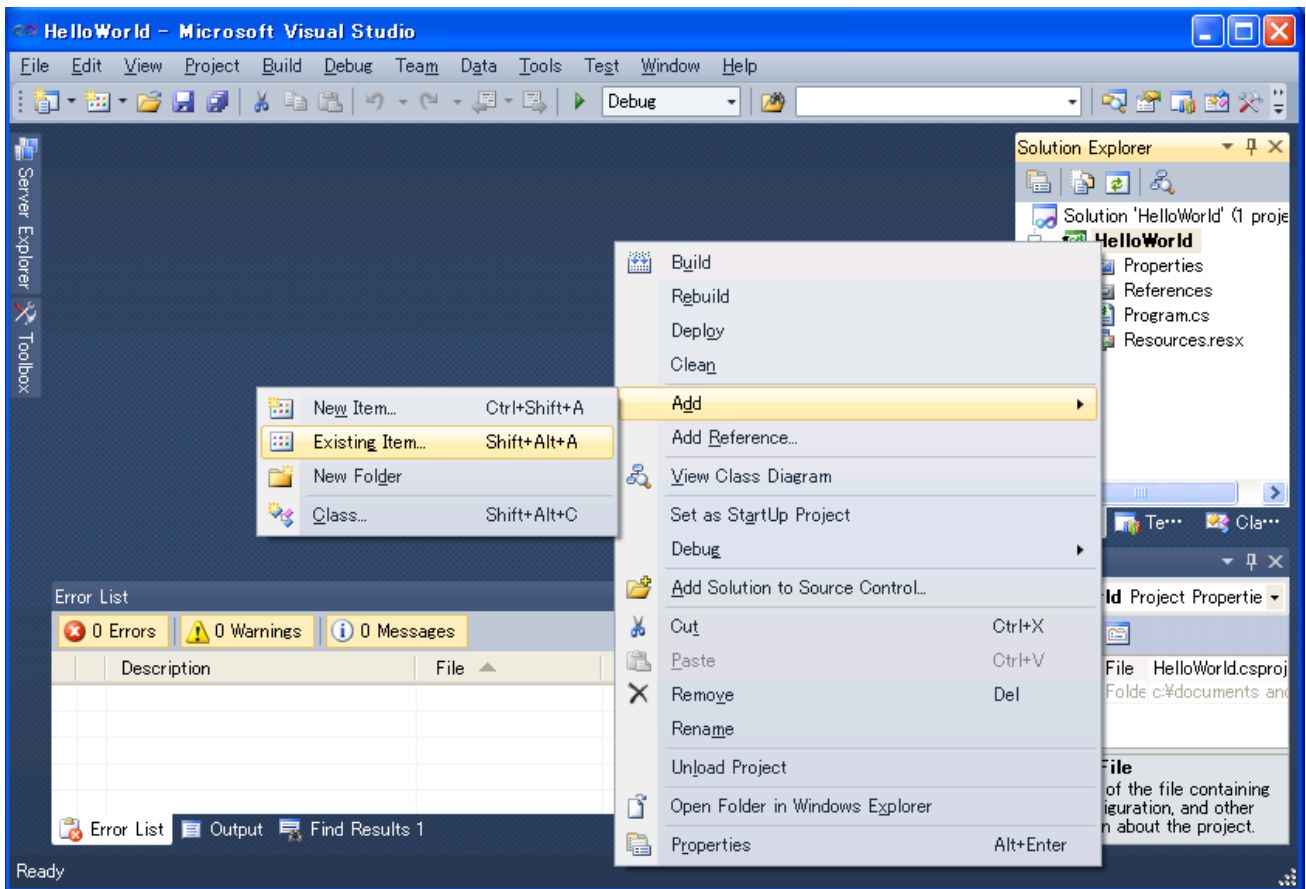


FIGURE 7.3 MICROSOFT VISUAL STUDIO

Select the file below:

C:\MicroFrameworkPK_v4_1\Product\Sample\HelloWorld\Main.cs

And press the Add button.

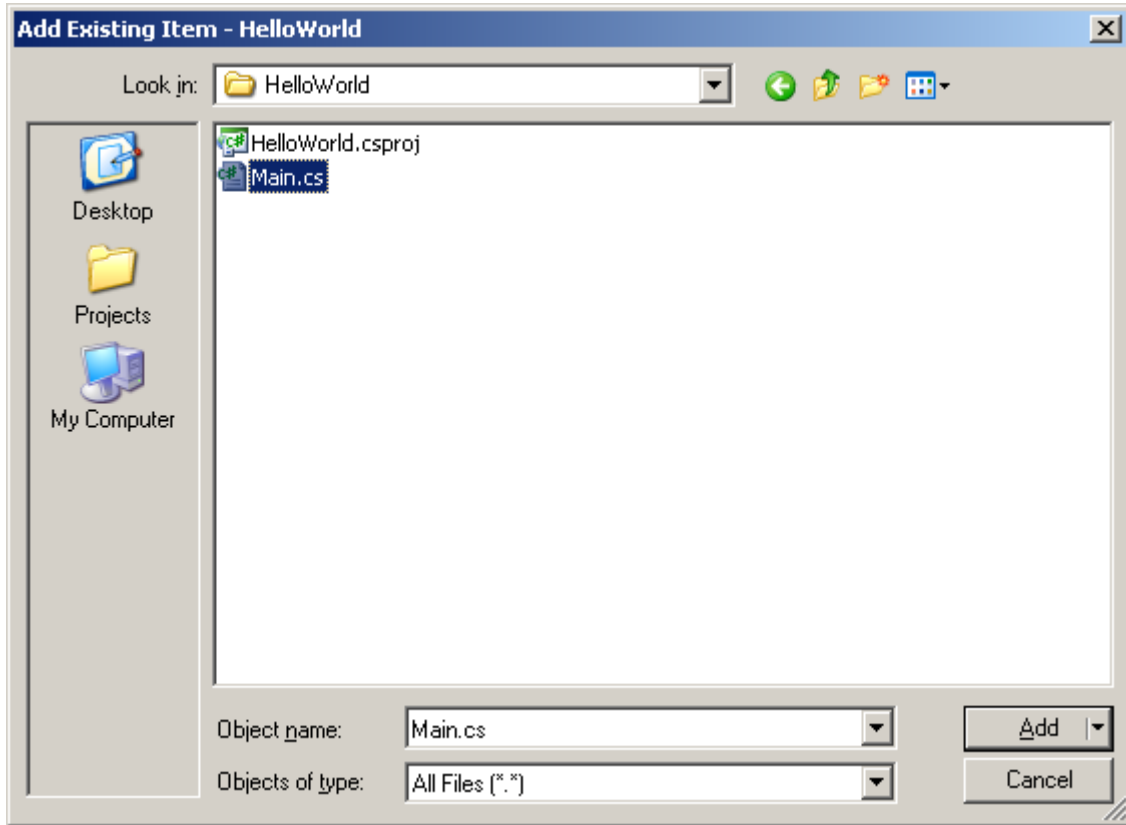


FIGURE 7.4 ADD EXISTING ITEM DIALOG BOX

iv) Remove the unnecessary file

Remove the Program.cs file from the HelloWorld project.

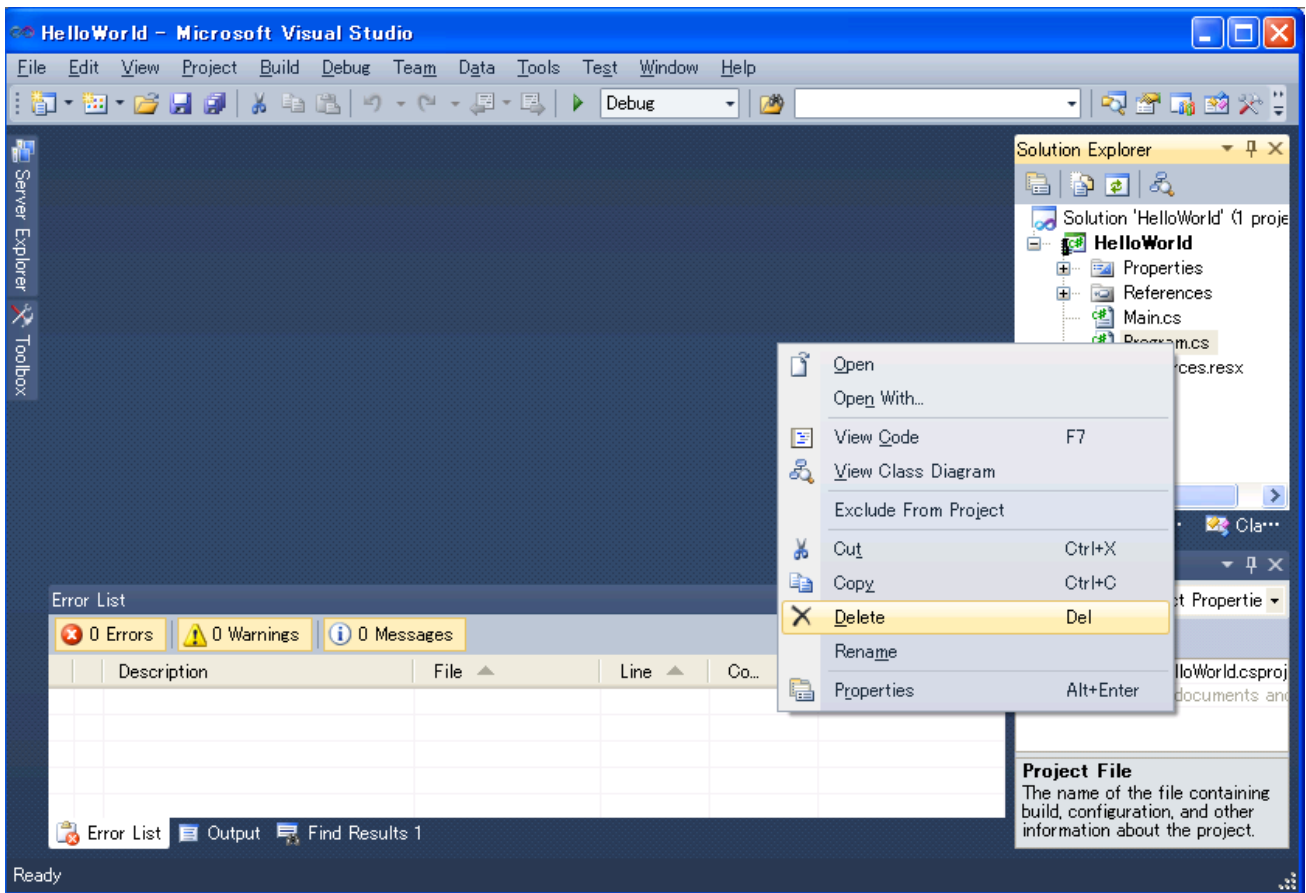


FIGURE 7.5 MICROSOFT VISUAL STUDIO

v) Build solution

Build the Solution by selecting the Build Solution menu,

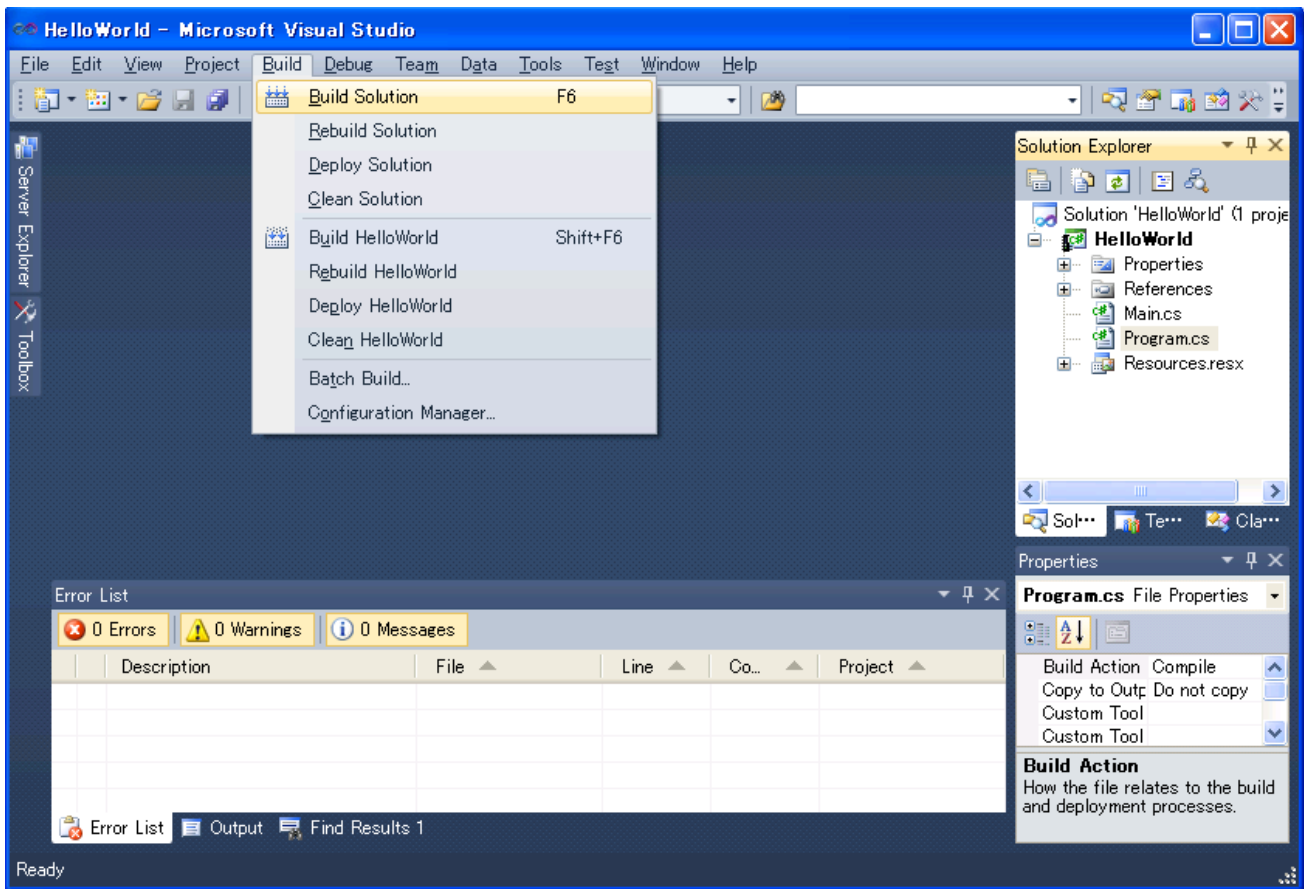


FIGURE 7.6 MICROSOFT VISUAL STUDIO

The build should be completed successfully.

DEPLOY AND RUN THE APPLICATION

In order for your board to communicate with Visual Studio, you need to download TinyCLR into flash memory on the board.

It is assumed that you have built TinyCLR and downloaded it into flash memory on the board at this point. If not, please build your Porting kit and download tinyclr.abs into flash memory as described in section 4.

i) Change properties for the current project.

Select Project menu -> HelloWorld Properties... to modify its properties.

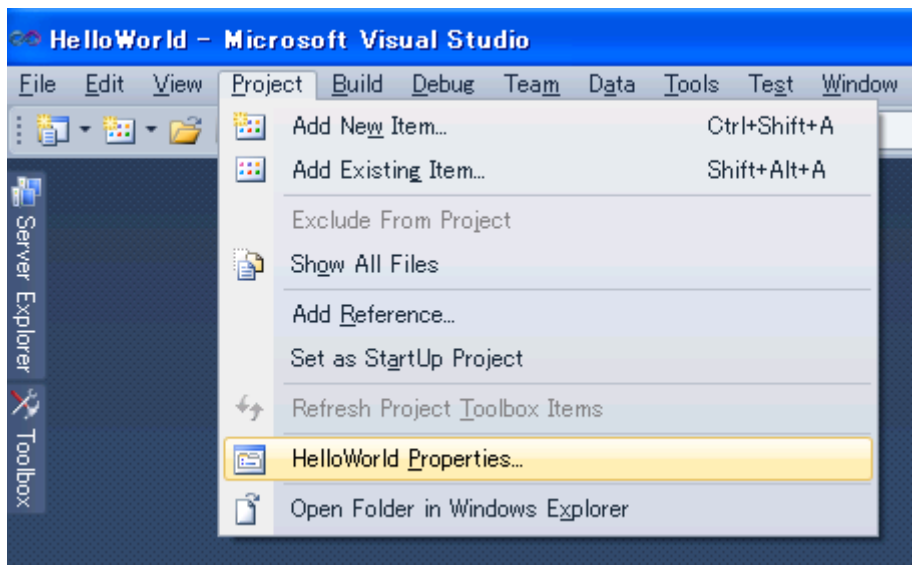


FIGURE 7.7 MICROSOFT VISUAL STUDIO

Select the “.NET Micro Framework” sheet and change the Deployment Transport to “Serial” as shown below, and then save and close this window.

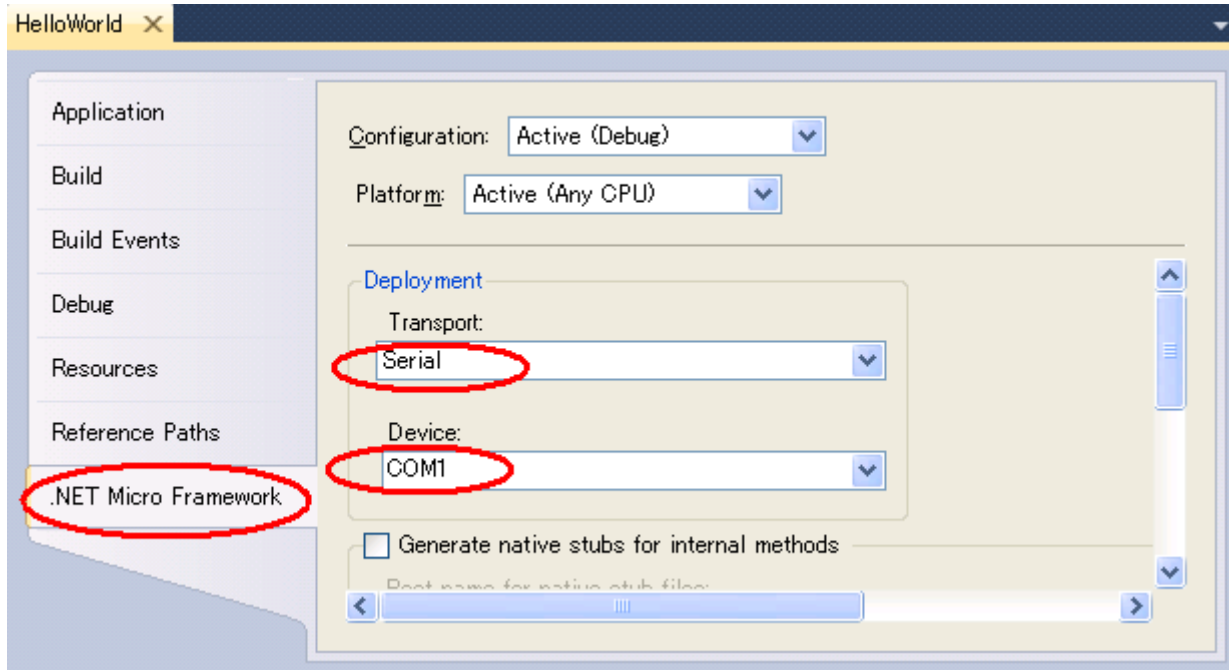


FIGURE 7.8 PROPERTIES...

ii) Rebuild

In order for these changes to take effect, a “Rebuild Solution” is required.

iii) Turn on the board

First, connect a null modem serial cable between your PC and the board and then apply power to the board.

iv) Deploying application

Select Build menu -> Deploy Solution.

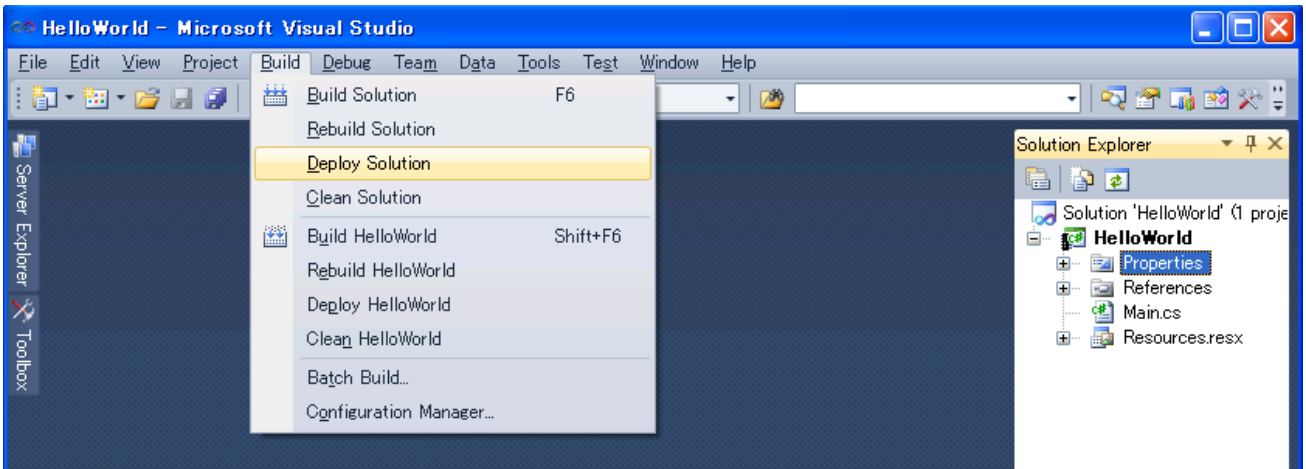


FIGURE 7.9 MICROSOFT VISUAL STUDIO

The message below is displayed in the output window at the start of the deployment process.

“Incrementally deploying assemblies to device”

When deployment finishes without problems, you will see the message below.

“Assemblies successfully deployed to device.”

If you have any trouble, please check that you have used the correct flash memory configuration file.

v) Run the application

Select Debug menu -> Start Debugging so that you can see the application running and use break point via Visual Studio.

If the deployment fails, it might be caused by unexpected data in the Deployment area in Flash memory. One way to avoid this is to erase all data in the deployment area using MFDeploy.exe. Section 9.3 describes how to use MFDeploy.exe.

HOW TO INCLUDE YOUR APPLICATION IN TINYCLR

The application can be included into tinyclr.abs. The simplest way to do this is just to add the PE files to your TinyCLR.proj file.

1) Build a generic application in the Porting Kit

Change the current folder to "C:\MicroFrameworkPK_v4_1" and execute the command below:

```
MSBUILD.EXE build.dirproj
```

2) Build an application without Visual Studio 2010.

Build your application using following command,

```
MSBUILD.EXE build.dirproj
```

For example, if you want to build the HelloWorld Sample application, use the following commands:

Change the current folder to "C:\MicroFrameworkPK_v4_1\Product\Sample"

and execute the command below:

```
MSBUILD.EXE build.dirproj
```

3) How to include your application into tinyclr.abs

Please add the following red lines after the property section of the TinyCLR.proj file. You will first have to make sure the .pe files have been built.

...

Change these lines as appropriate for your application

```
<Import Condition="" Project="$(SPOCLIENT)\Framework\Features\Diagnostics.featureproj" />
<Import Condition="" Project="$(SPOCLIENT)\Framework\Features\Core.featureproj" />
<Import Condition="" Project="$(SPOCLIENT)\Framework\Features\Serialization.featureproj" />

<ItemGroup>
<MMP_DAT_CreateDatabase Include="$(BUILD_TREE_CLIENT)\pe\$(ENDIANNESS)\mscorlib.pe"/>
<MMP_DAT_CreateDatabase Include="$(BUILD_TREE_CLIENT)\pe\$(ENDIANNESS)\Microsoft.SPOT.Native.pe"/>
<MMP_DAT_CreateDatabase Include="$(BUILD_TREE_CLIENT)\pe\$(ENDIANNESS)\Microsoft.SPOT.Net.pe"/>
<MMP_DAT_CreateDatabase Include="$(BUILD_TREE_CLIENT)\pe\$(ENDIANNESS)\Microsoft.SPOT.HelloWorld.pe"/>
<MMP_DAT_CreateDatabase Include="$(BUILD_TREE_CLIENT)\pe\$(ENDIANNESS)\System.pe"/>
<MMP_DAT_CreateDatabase Include="$(BUILD_TREE_CLIENT)\pe\$(ENDIANNESS)\Microsoft.SPOT.Graphics.pe"/>
<MMP_DAT_CreateDatabase Include="$(BUILD_TREE_CLIENT)\pe\$(ENDIANNESS)\Microsoft.SPOT.TinyCore.pe"/>
<MMP_DAT_CreateDatabase Include="$(BUILD_TREE_CLIENT)\pe\$(ENDIANNESS)\Microsoft.SPOT.Hardware.pe"/>
</ItemGroup>

<Import Project="$(SPOCLIENT)\tools\targets\Microsoft.SPOT.System.Interop.Settings" />
```

MFDEPLOY TOOL

In the Porting Kit, there is a useful Tool; MFDeploy.exe. Using this tool, you can make sure that TinyCLR works fine, examine the Flash memory mapping information, and erase the data in the Deployment area.

HOW TO BUILD AND RUN MFDEPLOY.EXE

Build and run MFDeploy.exe as follows:

1) Make sure the Timer driver, Power driver and the Serial driver work fine

Before using MFDeploy.exe, please make sure that the Timer driver, Power driver and the Serial driver work fine by using NativeSample.

2) Build MFDeploy.exe

You can build MFDeploy.exe with the command “MSBUILD.EXE build.dirproj” under the C:\MicroFrameworkPK_v4_1 folder.

Using this command, you can build not only MFDeploy.exe but also all of the managed tests in the PK as well.

MFDeploy is located in

\BuildOutput\public\Debug\Server\dll\MFDeploy.exe.

3) Run the MFDeploy.exe

i) Connect the COM Port to the board

Before running MFDeploy.exe, please make sure that the COM port in your PC is not being used by another application like Tera Term.

Then connect the COM port in your PC to the Serial connector on the board with a null modem cable.

ii) Run MFDeploy.exe

Run MFDeploy.exe by double-clicking it.

iii) Select the COM Port

Select the COM Port that you want to use for MFDeploy as shown below (List Box).

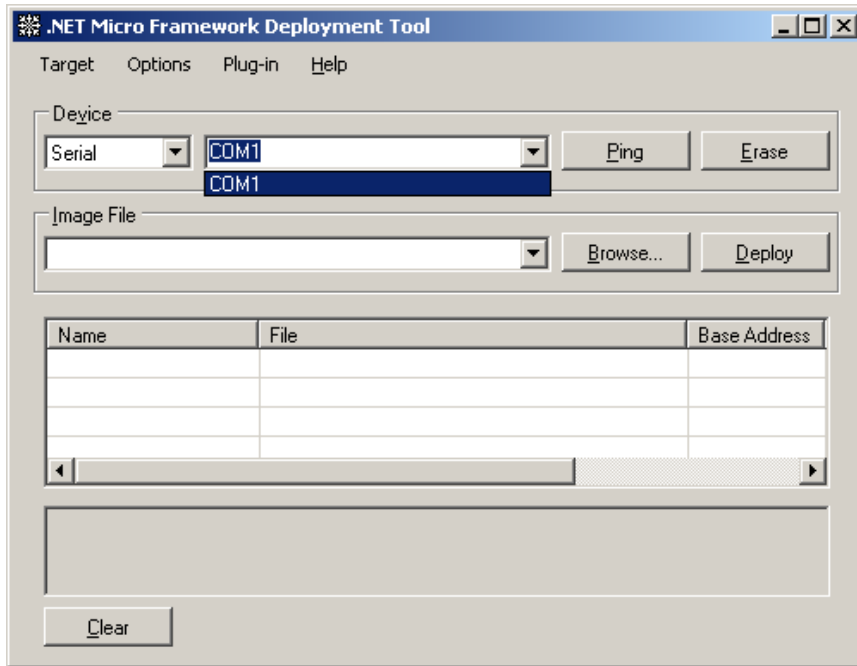


FIGURE 9.1 .NET MICRO FRAMEWORK DEPLOYMENT TOOL

iii) Connect the COM Port

Connect MFdeploy.exe to the COM Port by selecting the Target menu -> Connect option as shown below.

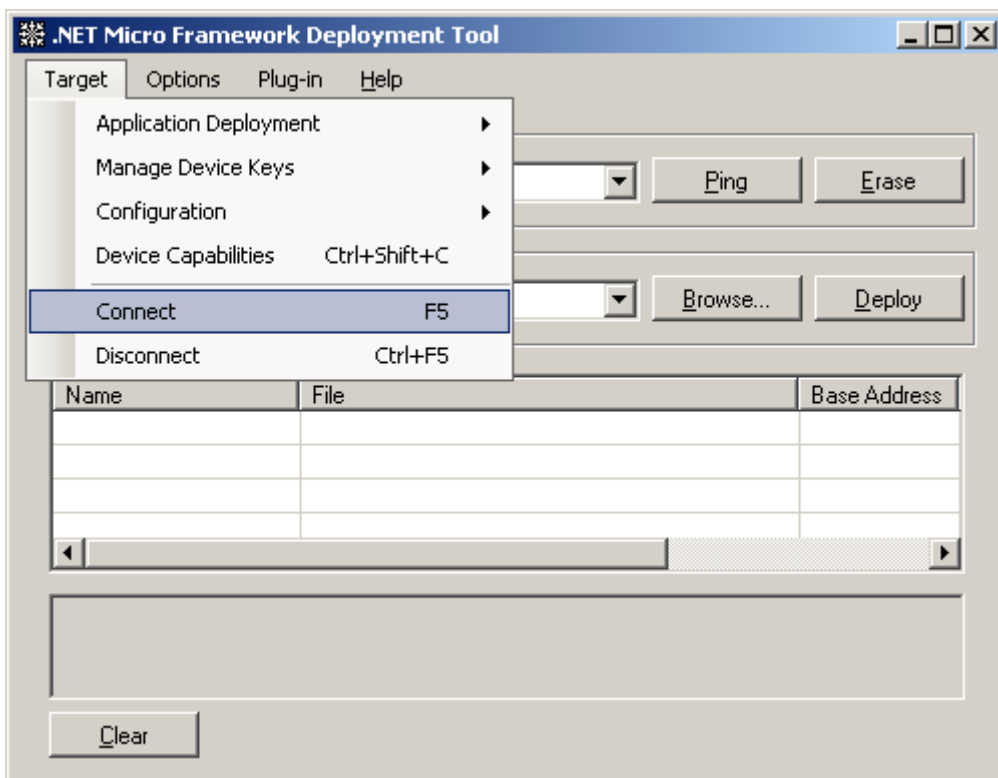


FIGURE 9.2 .NET MICRO FRAMEWORK DEPLOYMENT TOOL

HOW TO MAKE SURE IF TINYCLR WORKS FINE

If you cannot deploy the application properly, “MFDeploy.exe” is a good tool to make sure that TinyCLR works fine.

Turn on the board to see log output like below:

```
Connecting to COM1...Connected

TinyCLR (Build 4.0.2037.0)
Starting...
Created EE.
Started Hardware.
No debugger!
Create TS.
Loading start at 95000, end b5f88
Attaching file.
Assembly:mscorlib (4.0.2037.0) (3572 RAM - 29944 ROM - 17631 METADATA)
  AssemblyRef    =      0 bytes (      0 elements)
  TypeRef       =      0 bytes (      0 elements)
  FieldRef      =      0 bytes (      0 elements)
  MethodRef     =      0 bytes (      0 elements)
  TypeDef       =    1032 bytes (    129 elements)
  FieldDef      =     232 bytes (    115 elements)
  MethodDef     =    1448 bytes (    724 elements)

  Attributes    =      0 bytes (      0 elements)
  TypeSpec     =     16 bytes (      4 elements)
```

```
Resources = 232 bytes ( 29 elements)
Resources Files = 16 bytes ( 2 elements)
Resources Data = 437 bytes
Strings = 967 bytes
Signatures = 2015 bytes
ByteCode = 10500 bytes
```

Attaching file.

Assembly: Microsoft.SPOT.Native (4.0.2037.0) (1064 RAM - 5752 ROM - 4159 METADATA)

```
AssemblyRef = 4 bytes ( 1 elements)
TypeRef = 80 bytes ( 20 elements)
FieldRef = 0 bytes ( 0 elements)
MethodRef = 60 bytes ( 15 elements)
TypeDef = 328 bytes ( 41 elements)
FieldDef = 132 bytes ( 65 elements)
MethodDef = 216 bytes ( 108 elements)
```

```
Attributes = 48 bytes ( 6 elements)
TypeSpec = 0 bytes ( 0 elements)
Resources = 72 bytes ( 9 elements)
Resources Files = 8 bytes ( 1 elements)
Resources Data = 747 bytes
Strings = 207 bytes
Signatures = 587 bytes
ByteCode = 413 bytes
```

Attaching file.

Assembly: Microsoft.SPOT.Hardware (4.0.2037.0) (1752 RAM - 11404 ROM - 7365 METADATA)

```
AssemblyRef = 8 bytes ( 2 elements)
TypeRef = 124 bytes ( 31 elements)
FieldRef = 24 bytes ( 6 elements)
MethodRef = 120 bytes ( 30 elements)
TypeDef = 496 bytes ( 62 elements)
FieldDef = 176 bytes ( 88 elements)
MethodDef = 444 bytes ( 222 elements)
```

```
Attributes = 0 bytes ( 0 elements)
TypeSpec = 0 bytes ( 0 elements)
Resources = 0 bytes ( 0 elements)
Resources Files = 0 bytes ( 0 elements)
Resources Data = 0 bytes
Strings = 1329 bytes
Signatures = 1061 bytes
ByteCode = 2579 bytes
```

Attaching file.

Assembly: Microsoft.SPOT.Hardware.SerialPort (4.0.2037.0) (508 RAM - 3440 ROM - 1527 METADATA)

```
AssemblyRef = 8 bytes ( 2 elements)
TypeRef = 96 bytes ( 24 elements)
FieldRef = 0 bytes ( 0 elements)
MethodRef = 80 bytes ( 20 elements)
TypeDef = 16 bytes ( 2 elements)
FieldDef = 32 bytes ( 16 elements)
MethodDef = 92 bytes ( 46 elements)
```

```
Attributes = 0 bytes ( 0 elements)
TypeSpec = 0 bytes ( 0 elements)
Resources = 0 bytes ( 0 elements)
Resources Files = 0 bytes ( 0 elements)
Resources Data = 0 bytes
Strings = 667 bytes
Signatures = 239 bytes
ByteCode = 1118 bytes
```

Attaching file.

Assembly: Microsoft.SPOT.IO (4.0.2037.0) (716 RAM - 4432 ROM - 2459 METADATA)

```
AssemblyRef = 12 bytes ( 3 elements)
TypeRef = 72 bytes ( 18 elements)
FieldRef = 0 bytes ( 0 elements)
MethodRef = 96 bytes ( 24 elements)
TypeDef = 120 bytes ( 15 elements)
FieldDef = 68 bytes ( 34 elements)
MethodDef = 140 bytes ( 70 elements)
```

```
Attributes      =          0 bytes (          0 elements)
TypeSpec        =          0 bytes (          0 elements)
Resources       =          0 bytes (          0 elements)
Resources Files =          0 bytes (          0 elements)
Resources Data  =          0 bytes
Strings         =         646 bytes
Signatures      =         335 bytes
ByteCode        =        1199 bytes
```

Attaching file.

Assembly: System.IO (4.0.2037.0) (1548 RAM - 13264 ROM - 5862 METADATA)

```
AssemblyRef     =          8 bytes (          2 elements)
TypeRef         =        168 bytes (         42 elements)
FieldRef        =          36 bytes (          9 elements)
MethodRef       =        392 bytes (         98 elements)
TypeDef         =        144 bytes (         18 elements)
FieldDef        =          76 bytes (         37 elements)
MethodDef       =        392 bytes (        195 elements)
```

```
Attributes      =          0 bytes (          0 elements)
TypeSpec        =          8 bytes (          2 elements)
Resources       =          0 bytes (          0 elements)
Resources Files =          0 bytes (          0 elements)
Resources Data  =          0 bytes
Strings         =        356 bytes
Signatures      =         790 bytes
ByteCode        =        6919 bytes
```

Attaching file.

Assembly: Microsoft.SPOT.Graphics (4.0.2037.0) (388 RAM - 2268 ROM - 1357 METADATA)

```
AssemblyRef     =          8 bytes (          2 elements)
TypeRef         =         24 bytes (          6 elements)
FieldRef        =          0 bytes (          0 elements)
MethodRef       =         20 bytes (          5 elements)
TypeDef         =         40 bytes (          5 elements)
FieldDef        =         16 bytes (          8 elements)
MethodDef       =         96 bytes (         48 elements)
```

```
Attributes      =          0 bytes (          0 elements)
TypeSpec        =          0 bytes (          0 elements)
Resources       =          0 bytes (          0 elements)
Resources Files =          0 bytes (          0 elements)
Resources Data  =          0 bytes
Strings         =         537 bytes
Signatures      =         293 bytes
ByteCode        =         242 bytes
```

Attaching file.

Assembly: Microsoft.SPOT.TinyCore (4.0.2037.0) (5080 RAM - 61564 ROM - 23446 METADATA)

```
AssemblyRef     =         16 bytes (          4 elements)
TypeRef         =        224 bytes (         56 elements)
FieldRef        =         52 bytes (         13 elements)
MethodRef       =        456 bytes (        114 elements)
TypeDef         =       1104 bytes (        138 elements)
FieldDef        =        728 bytes (        363 elements)
MethodDef       =       1576 bytes (        787 elements)
```

```
Attributes      =          0 bytes (          0 elements)
TypeSpec        =          4 bytes (          1 elements)
Resources       =          0 bytes (          0 elements)
Resources Files =          0 bytes (          0 elements)
Resources Data  =          0 bytes
Strings         =       12916 bytes
Signatures      =        3122 bytes
ByteCode        =       25075 bytes
```

Attaching file.

Assembly: Microsoft.SPOT.Time (4.0.2037.0) (508 RAM - 2976 ROM - 1552 METADATA)

```
AssemblyRef     =         12 bytes (          3 elements)
TypeRef         =         60 bytes (         15 elements)
FieldRef        =          0 bytes (          0 elements)
MethodRef       =         36 bytes (          9 elements)
TypeDef         =         96 bytes (         12 elements)
FieldDef        =         40 bytes (         20 elements)
MethodDef       =         84 bytes (         41 elements)
```

```
Attributes      =          0 bytes (          0 elements)
```

```
TypeSpec      =          0 bytes (          0 elements)
Resources     =          0 bytes (          0 elements)
Resources Files =          0 bytes (          0 elements)
Resources Data =          0 bytes
Strings       =        895 bytes
Signatures    =         220 bytes
ByteCode     =         403 bytes
```

Loading Deployment Assemblies.
Resolving.

```
Total: (12884 RAM - 135044 ROM - 65358 METADATA)
AssemblyRef   =          76 bytes (          19 elements)
TypeRef       =          848 bytes (         212 elements)
FieldRef      =          112 bytes (          28 elements)
MethodRef     =         1260 bytes (         315 elements)
TypeDef       =         3376 bytes (         422 elements)
FieldDef      =         1500 bytes (         746 elements)
MethodDef     =         4488 bytes (        2241 elements)

DebuggingInfo =         2252 bytes

Attributes    =          48 bytes (          6 elements)
TypeSpec     =          28 bytes (          7 elements)
Resources Files =          72 bytes (          3 elements)
Resources     =          304 bytes (         38 elements)
Resources Data =         1184 bytes
Strings       =       18520 bytes
Signatures    =        8662 bytes
ByteCode     =       48448 bytes
```

```
GC: 1msec 15276 bytes used, 4178820 bytes available
Type 0F (STRING          ):          24 bytes
Type 15 (FREEBLOCK      ): 4178820 bytes
Type 17 (ASSEMBLY       ):    15180 bytes
Type 34 (APPDOMAIN_HEAD ):          72 bytes
```

```
Total: (12884 RAM - 135044 ROM - 65358 METADATA)
AssemblyRef   =          76 bytes (          19 elements)
TypeRef       =          848 bytes (         212 elements)
FieldRef      =          112 bytes (          28 elements)
MethodRef     =         1260 bytes (         315 elements)
TypeDef       =         3376 bytes (         422 elements)
FieldDef      =         1500 bytes (         746 elements)
MethodDef     =         4488 bytes (        2241 elements)

DebuggingInfo =         2252 bytes

Attributes    =          48 bytes (          6 elements)
TypeSpec     =          28 bytes (          7 elements)
Resources Files =          72 bytes (          3 elements)
Resources     =          304 bytes (         38 elements)
Resources Data =         1184 bytes
Strings       =       18520 bytes
Signatures    =        8662 bytes
ByteCode     =       48448 bytes
```

```
Ready.
Cannot find any entrypoint!
Done.
Waiting for debug commands...
```

After the above messages have appeared, you should try to connect by clicking the “Ping” button in MFdeploy. If you see that TinyCLR responds, then it means that TinyCLR is up and running. After performing the erase operation, you may need to reset or cycle power to the SH7619 EVB.

Pinging... TinyCLR

ERASE DATA IN THE DEPLOYMENT AREA

Following is the way to erase data in the deployment area.

- 1) Connect the Target Platform by referring to section 9.1.
- 2) Press the Erase button to start the Erase operation.

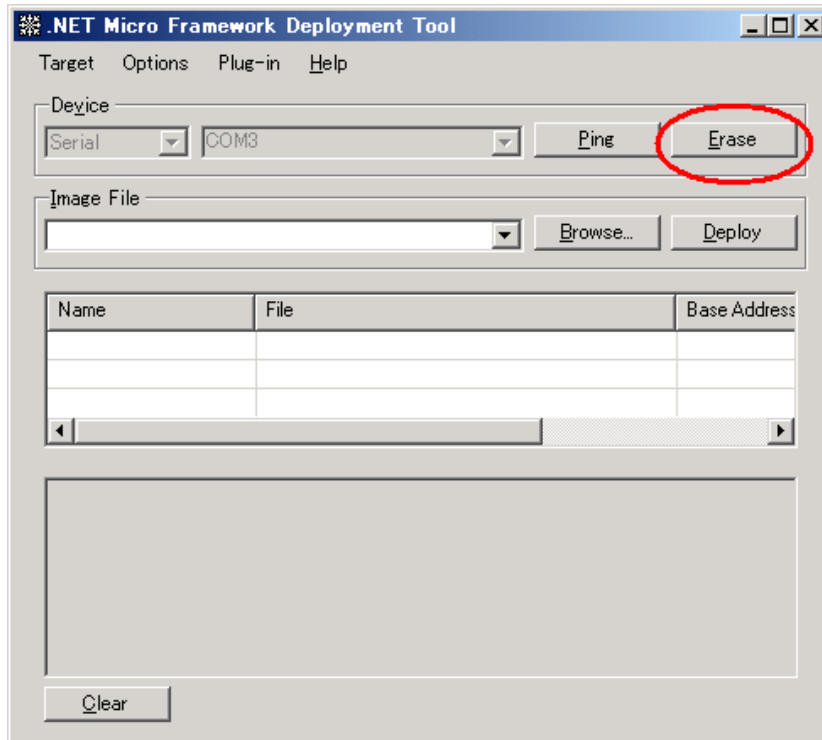


FIGURE 9.3 .NET MICRO FRAMEWORK DEPLOYMENT TOOL

- 3) If the error shown below is displayed, that's OK.



FIGURE 9.4 ERROR MESSAGE

4) The following messages show up when data in the deployment area has erased.

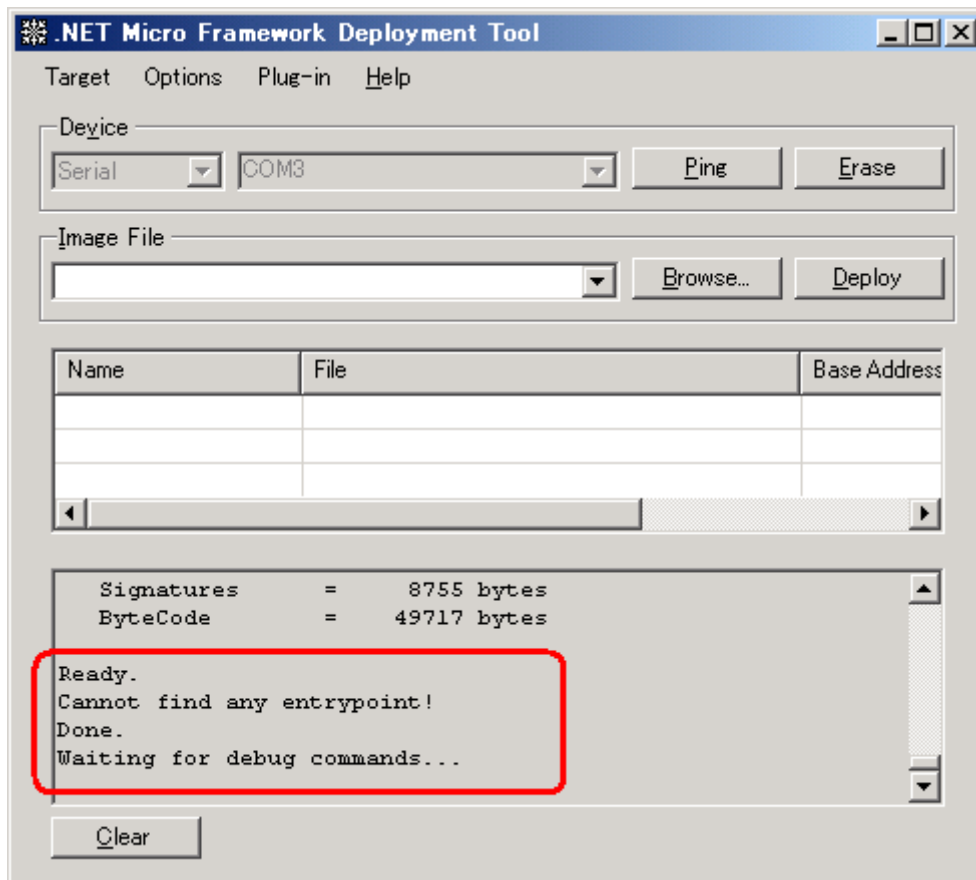


FIGURE 9.5 .NET MICRO FRAMEWORK DEPLOYMENT TOOL

WIRING

This section describes the wiring for the existing LCD and Keypad driver hardware.

1) Wiring for the LCD panel

The 132 x 176 TFT LCD Display (HD66773R) is interfaced to area CS6B of the SH7619 Bus State Controller (BSC). 3.3V power is provided by the SH7619 board, while 5V power is provided by a separate DC power supply connected to the LCD Expansion board. A 40-pin header connector is used for the LCD module interface; the pin assignment for the 40-pin header is shown below:

Pin No.	Signal Name	I/O	Pin No.	Signal Name	I/O
1	GND	—	21	D3	I/O
2	/RESET	OUT	22	D2	I/O
3	3.3V supply	—	23	GND	—
4	NC	—	24	D1	I/O
5	NC	—	25	D0	I/O
6	D15	I/O	26	GND	—
7	D14	I/O	27	/LCD-RD	OUT
8	GND	—	28	/LCD-WR	OUT
9	D13	I/O	29	LCD-RS1	OUT
10	D12	I/O	30	LCD-RS2	OUT
11	D11	I/O	31	GND	—
12	D10	I/O	32	3.3V supply	—
13	GND	—	33	5V supply	—
14	D9	I/O	34	5V supply	—
15	D8	I/O	35	GND	—
16	D7	I/O	36	/LCD-CS1	OUT
17	D6	I/O	37	/LCD-CS2	OUT
18	GND	—	38	LCD-TMG1	IN
19	D5	I/O	39	LCD-TMG2	IN
20	D4	I/O	40	LCD-DIR	OUT

FIGURE 10.1 LCD HEADER PIN ASSIGNMENT

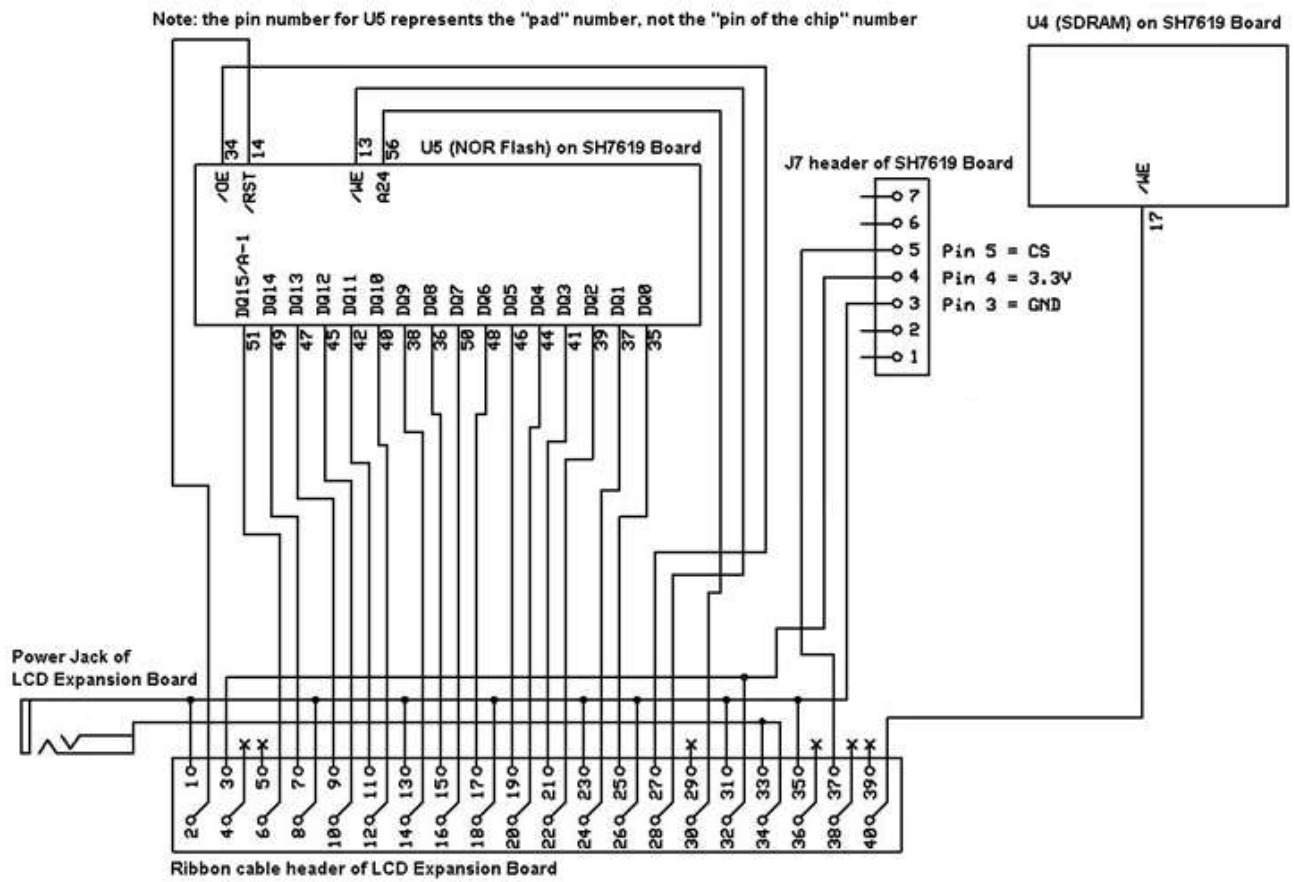


Figure 10.2 Schematic of the LCD Expansion Board interface

2) Wiring for the GPIO for Button Keypad

In this Porting Kit, it is assumed that a 25-button keypad matrix is connected to ten of the SH7619 GPIO signals. Five of these GPIOs (PC08, PC10, PC09, PC11, and PC13) should be configured for output. The other five GPIOs (PC00, PC01, PC02, PC03, and PC17) should be configured for input and they must have pull-up resistors; meaning the buttons are active low.

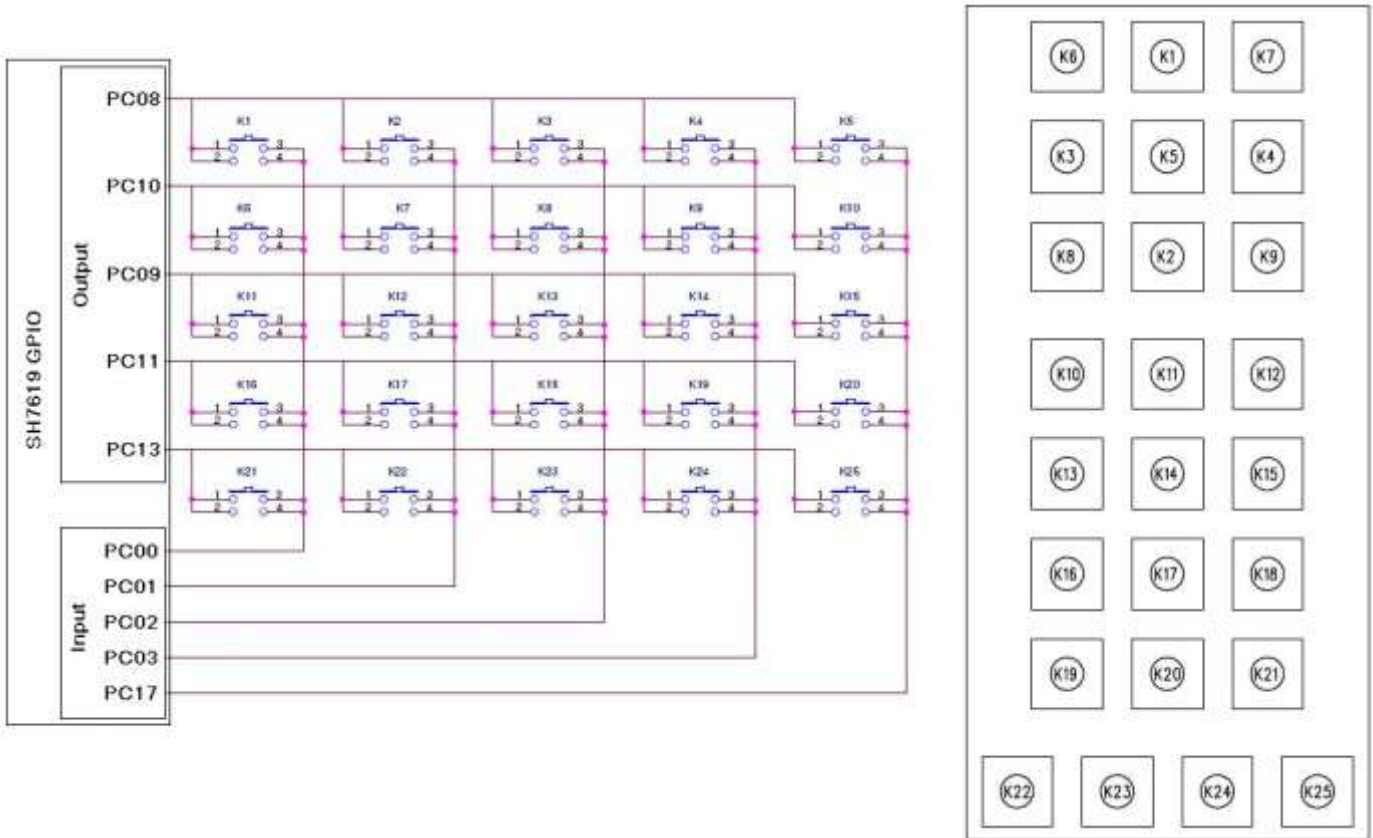


Figure 10.3 The GPIO button configuration and perspective orientation