

EMX Development System

Rev.1.10

July 21, 2010

Getting Started

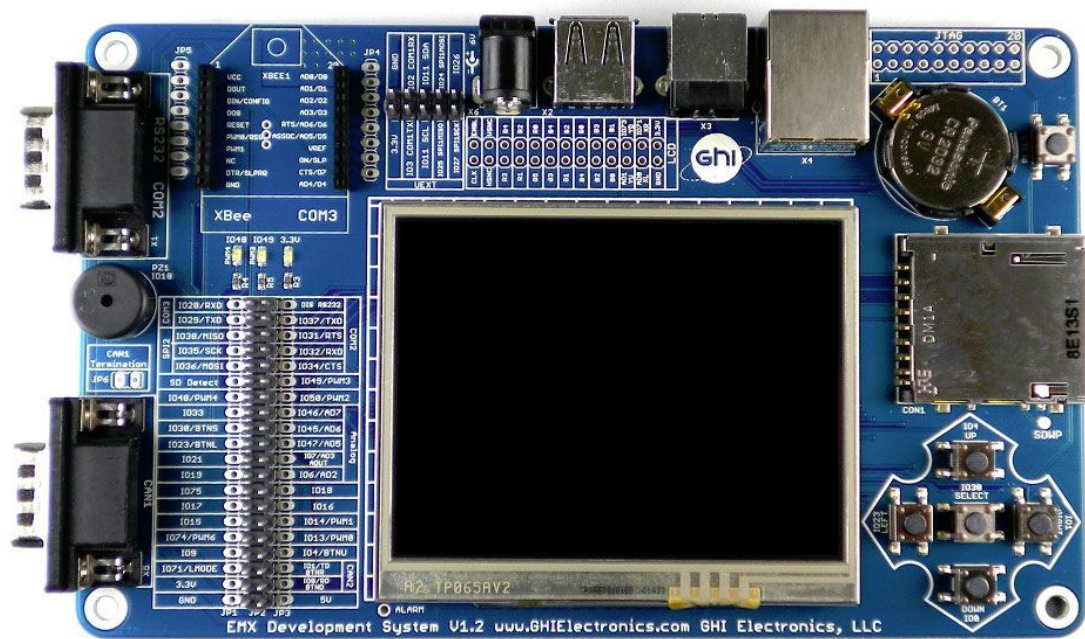


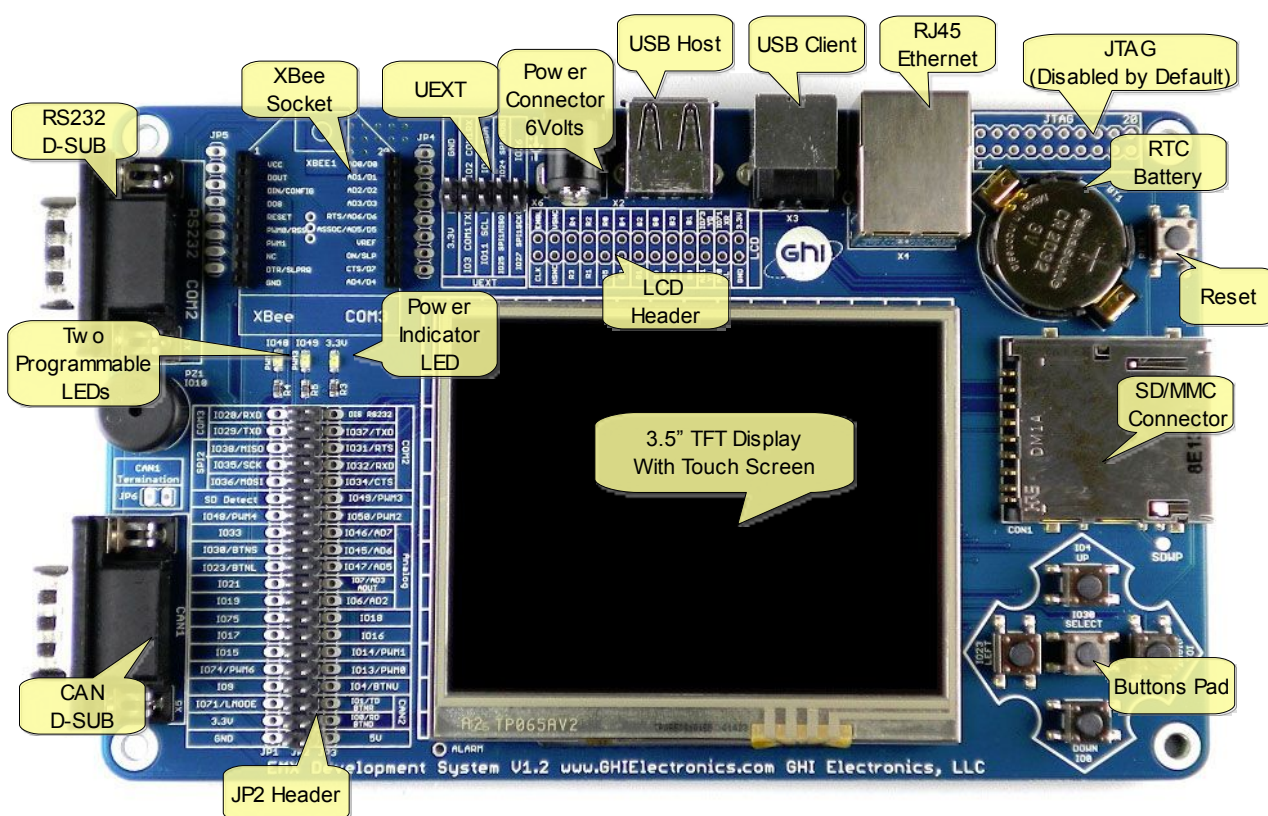
Table of Contents

1.Introduction.....	3
The objective of this Guide.....	4
2.Getting Started.....	5
2.1.System Setup.....	5
2.2.The Emulator.....	6
Create a Project.....	6
Selecting Transport Interface.....	8
Executing (Deploying) the project.....	9
Debugging Breakpoints.....	10
2.3.Running on Hardware.....	11
MFDeploy can Ping!.....	12
Deploying to Hardware.....	13
Connect MFDeploy to show debugging messages.....	13
Checking EMX firmware and TinyBooter Version.....	15
2.4.Deleting The Deployed Project.....	15
Emergency user application code deletion.....	15
3.Assemblies with EMX Project.....	16
3.1.What are Assemblies?.....	16
Standard Assemblies.....	16
GHI NETMF assemblies.....	16
Adding Assemblies to Visual C# Project.....	17
4.TinyBooter and Firmware Update.....	18
4.1.Checking the Firmware Version.....	18
4.2.TinyBooter Update through GHI Boot Loader.....	20
4.3.EMX firmware update through TinyBooter.....	23
5.What is next?.....	27
Legal Notice.....	28
Licensing.....	28
Disclaimer.....	28

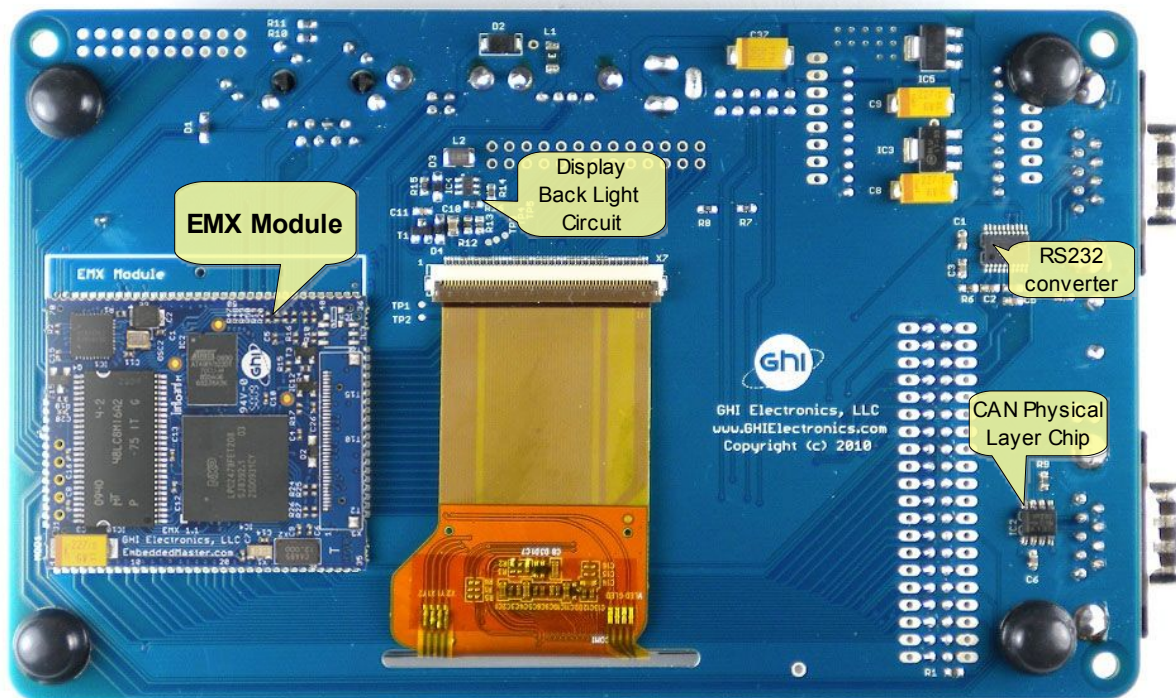
1. Introduction

EMX Development System is the official kit from GHI Electronics for the EMX module. This kit exposes the various peripherals and interfaces that make it an ideal starting point for any .NET Micro Framework project. Furthermore, most of EMX module signals such as GPIO, SPI and UART are accessible on a 0.1" header for rapid prototyping.

EMX Development System [Brochure](#) and [Pin-outs Document](#) provides for a more detailed view of this system.



Front View



Back View

The objective of this Guide

This guide will help you in the first steps to use EMX Module through the Development System. From here, we will show you how to connect EMX, make sure it is running, load a simple program, and explain how to update the internal firmware. This guide only covers very basic points about EMX. More details are provided in the user manual.

2. Getting Started

2.1. System Setup

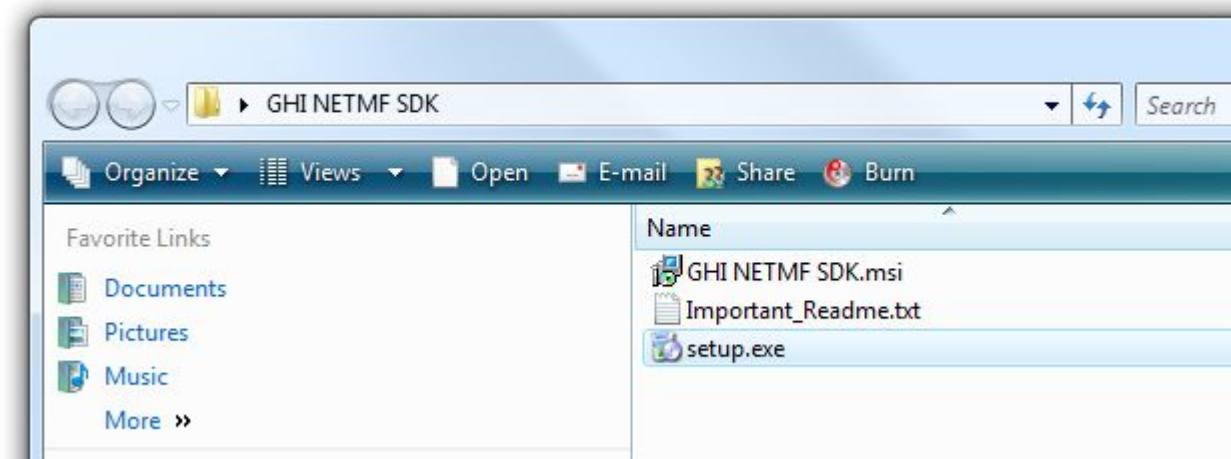
Before we try anything, we want to make sure the PC is setup with needed software. First download and install [Visual C# Express 2010](#).

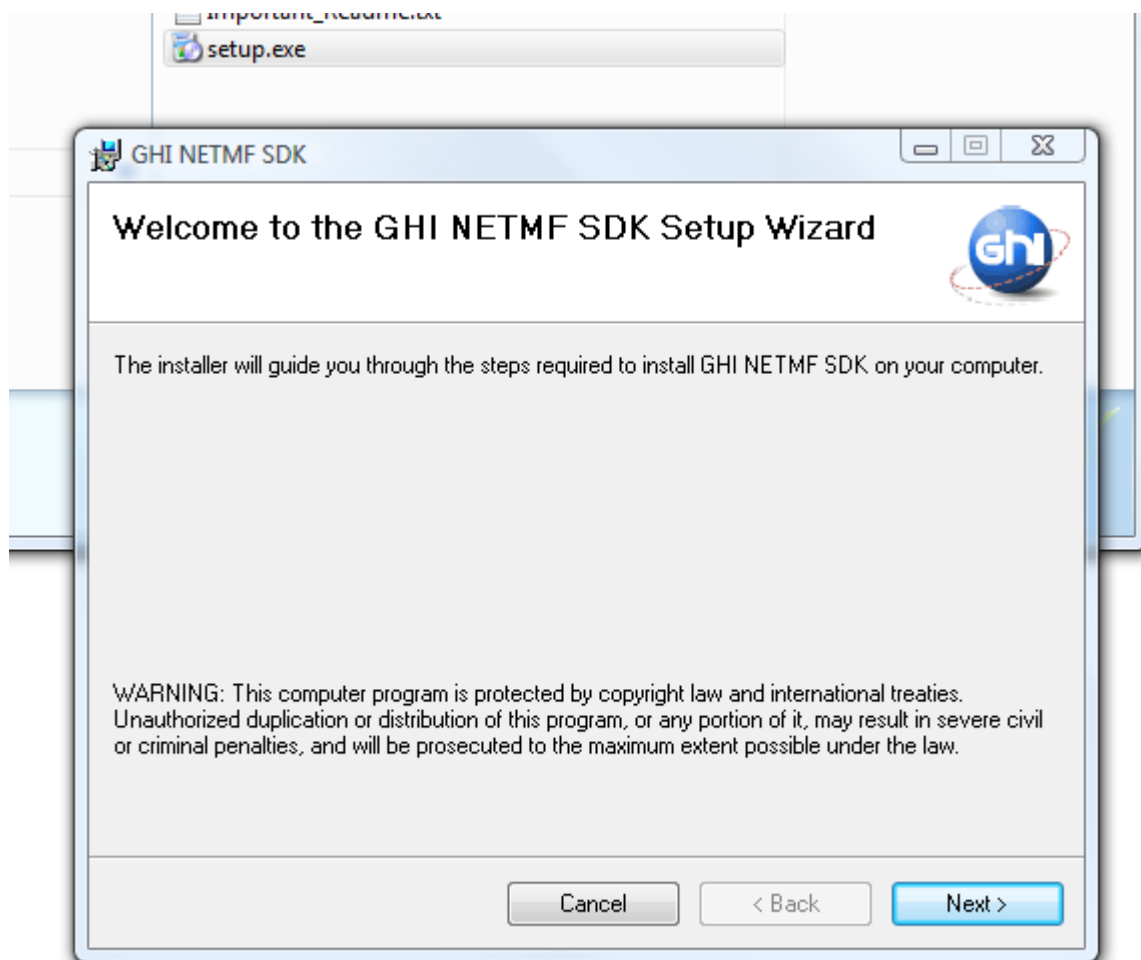
Now, download and install [.NET Micro Framework 4.1 SDK](#) (not the porting kit) If you have Visual Studio 2010 then you can skip this step.

if link above didn't work, search for ".NET Micro Framework 4.1 SDK".

Finally, install the GHI NETMF SDK (Software Development Kit) from EMX's downloads page.

The SDK comes in a zip file, extract it and then run setup.exe to install the SDK.



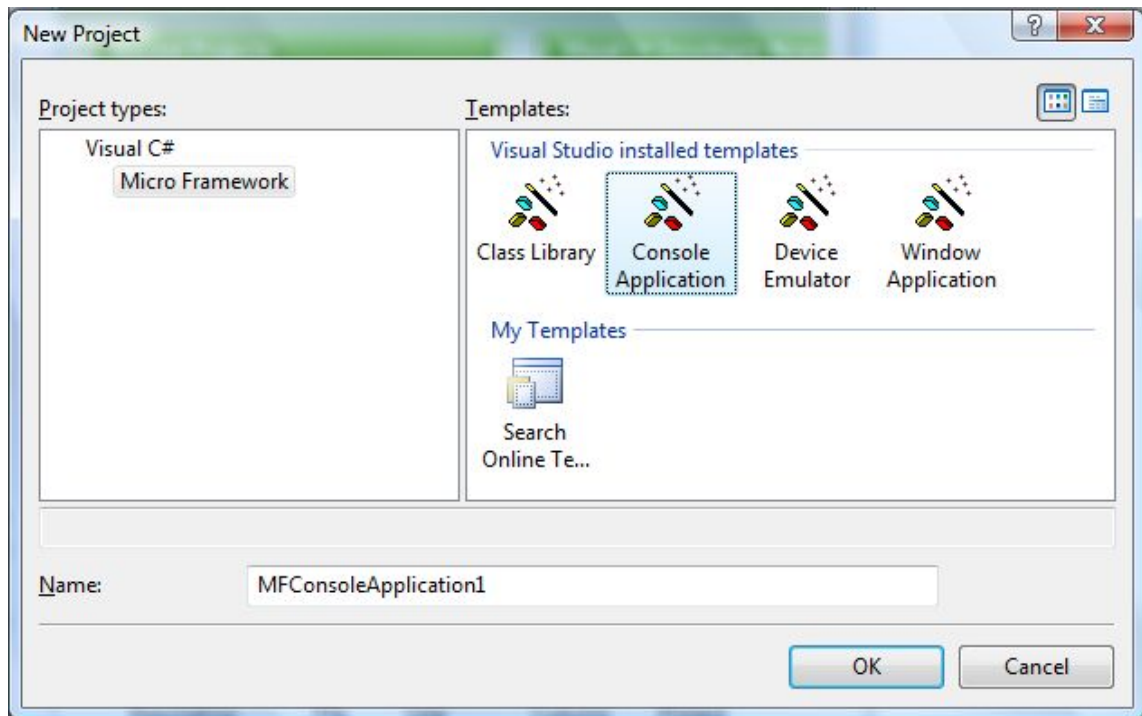


2.2. The Emulator

NETMF includes an emulator that allows running application right on the PC. For our first project, we will use the emulator to run a very simple application.

Create a Project

Open Visual C# express and, from the menu, select file -> New Project. The wizard now should have "Micro Framework" option in the left menu. Click on it, and from the templates, select "Console Application"



Click the “OK” button and you will have a new project that is ready to run. The project has only one C# file, called Program.cs, which contains a few lines of code. The file is shown in “Solution Explorer” window. If this window is not showing then you can open it by clicking “View->Solution Explorer” from the menu.

```
using System;
using Microsoft.SPOT;

namespace MFConsoleApplication1
{
    public class Program
    {
        public static void Main()
        {
            Debug.Print(
                Resources.GetString(Resources.StringResources.String1));
        }
    }
}
```

For simplicity change the code to make it look like the listing below.

```
using System;
using Microsoft.SPOT;

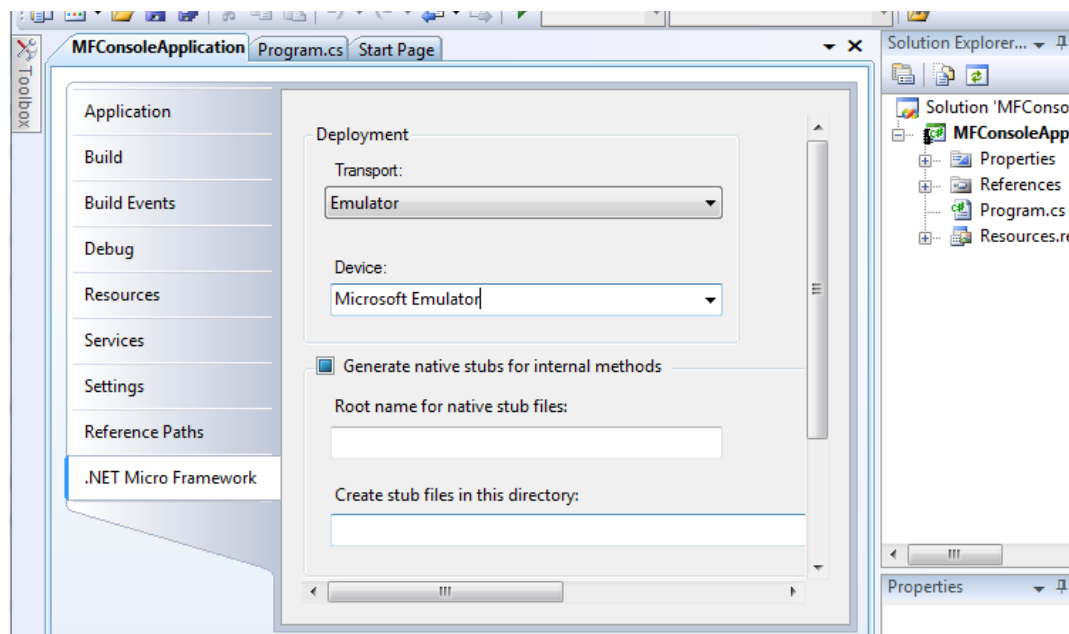
public class Program
{
    public static void Main()
    {
        Debug.Print("Amazing!");
    }
}
```

Selecting Transport Interface

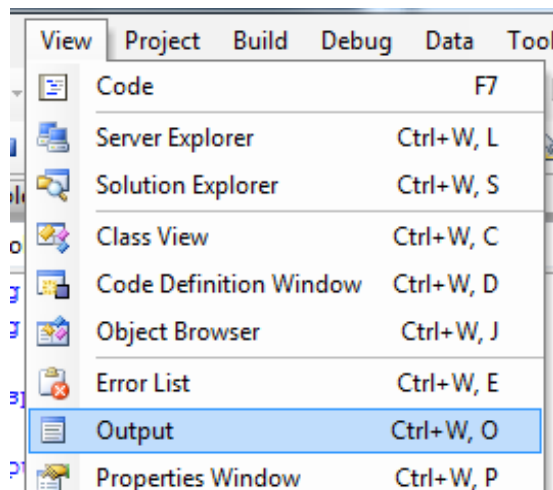
Do not worry if you do not understand the code. We will explain it later. For now, we want to run it on the emulator. Let us make sure you have everything setup properly. Click on “Project->Properties” from the menu. In the new showing window, we want to make sure we select the emulator. On the left side tabs, select “.NET Micro Framework” and make sure the window looks like the image below.

Transport: Emulator

Device: Microsoft Emulator



Make sure the output window is visible, click on “View->Output”

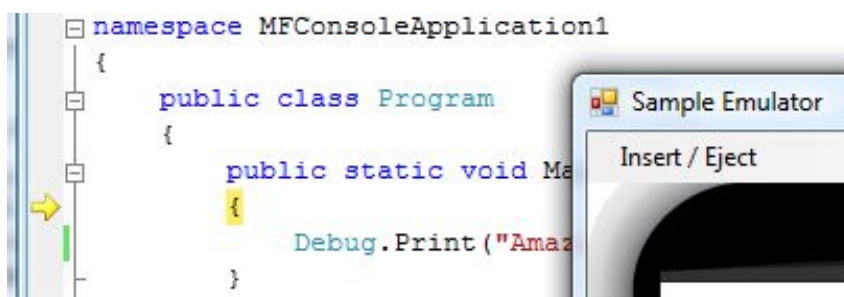


Executing (Deploying) the project

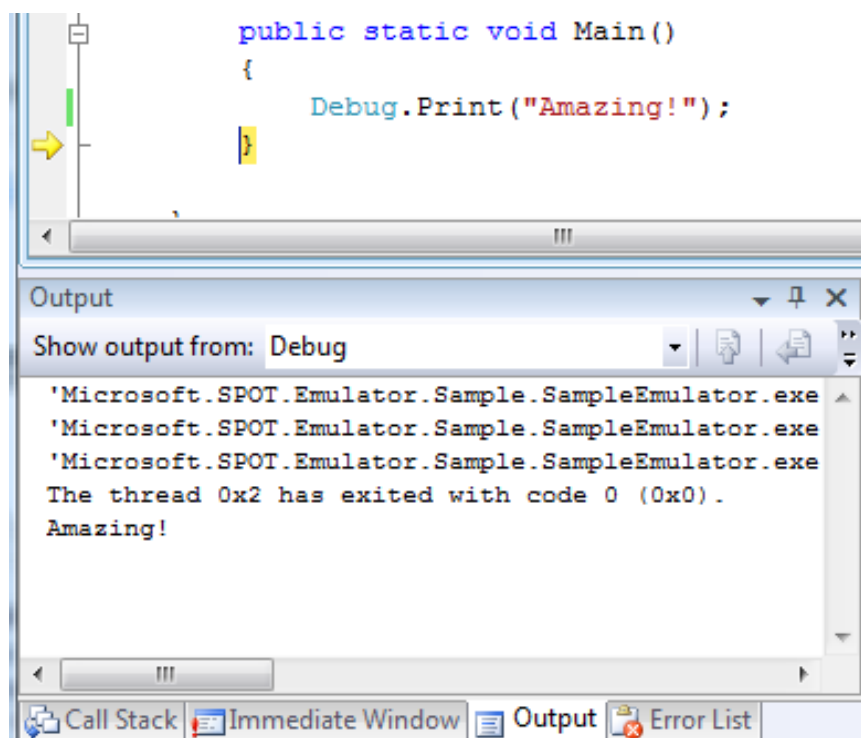
Finally, we are ready to run our first application. Press F5 key on the computer. This is a very useful shortcut and you will be using it a lot to run your applications. After you press F5, the application will be compiled and loaded on the emulator, and in couple seconds everything will stop! That is because our program had finished execution so fast that we didn't see much.

We want to “debug” the code now. Debugging means that you are able to step in the code and see what it is doing. This is one of the greatest values of NETMF.

This time use F11 instead of F5, this will “step” in the application instead of just running it. This will deploy the application on the emulator and stop at the very first line of the code. This is indicated by the yellow arrow.



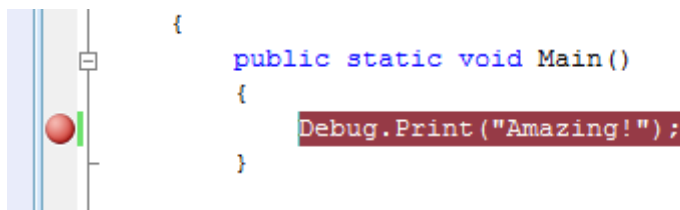
C# applications always start from a method called Main, and this is where the arrow had stopped. Press F11 again and the debugger will run the next line of code, which is the line you changed before. You probably have guessed it right, this line will print "Amazing!" to the debug window. The debug window is the output window on Visual C# express. Make sure Output window is visible like explained earlier and press F11 one more time. Once you step on that line, you will see the word Amazing! Showing in the output window.



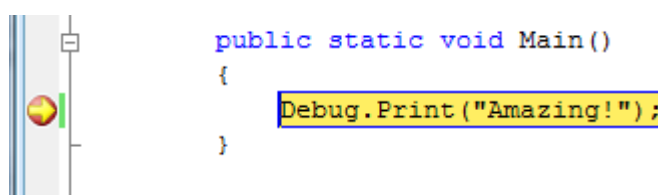
The Program that you've just deployed is called "Managed Code" in .NET Micro Framework Terminology.

Debugging Breakpoints

Breakpoints are another useful feature when debugging code. While the application is running, the debugger checks if execution has reached a breakpoint. If so, the execution will pause. Click the bar right to the left of the line that prints "Amazing!". This will show a red dot which is the breakpoint.



Now press F5 to run the software and when the application reaches the breakpoint the debugger will pause it as showing in the image below.

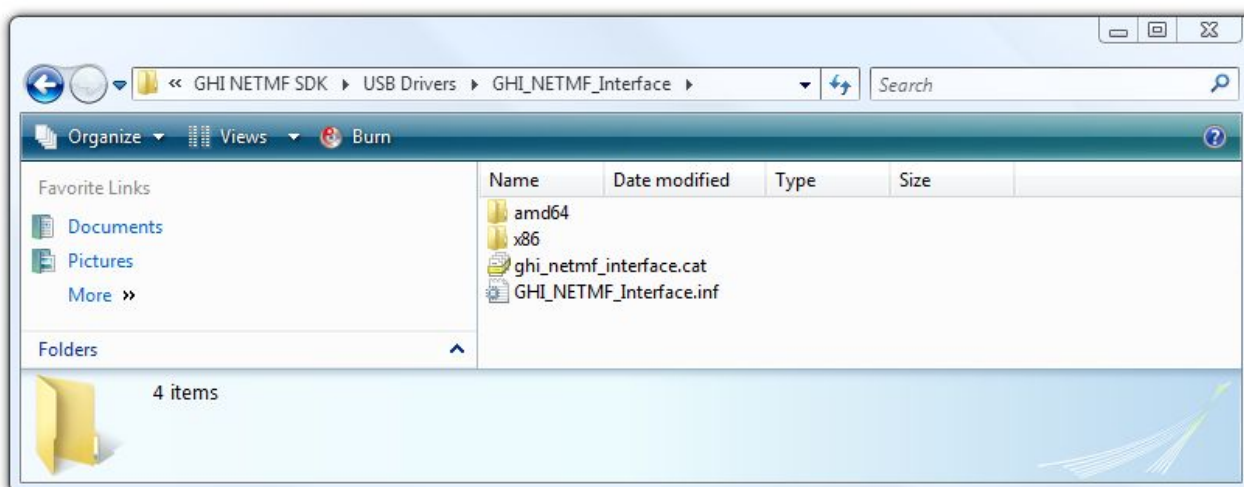


Now, you can step in the code using F11 or continue execution using F5.

2.3. Running on Hardware

We are now ready to connect EMX Development System to the PC. Connect The development system using the USB cable. No power is needed as EMX Development System uses USB for power.

If this is the first time you plug the system to your PC then windows will ask for drivers. Direct windows to the GHI NETMF interface driver available in GHI NETMF SDK under USB Drivers Folder.

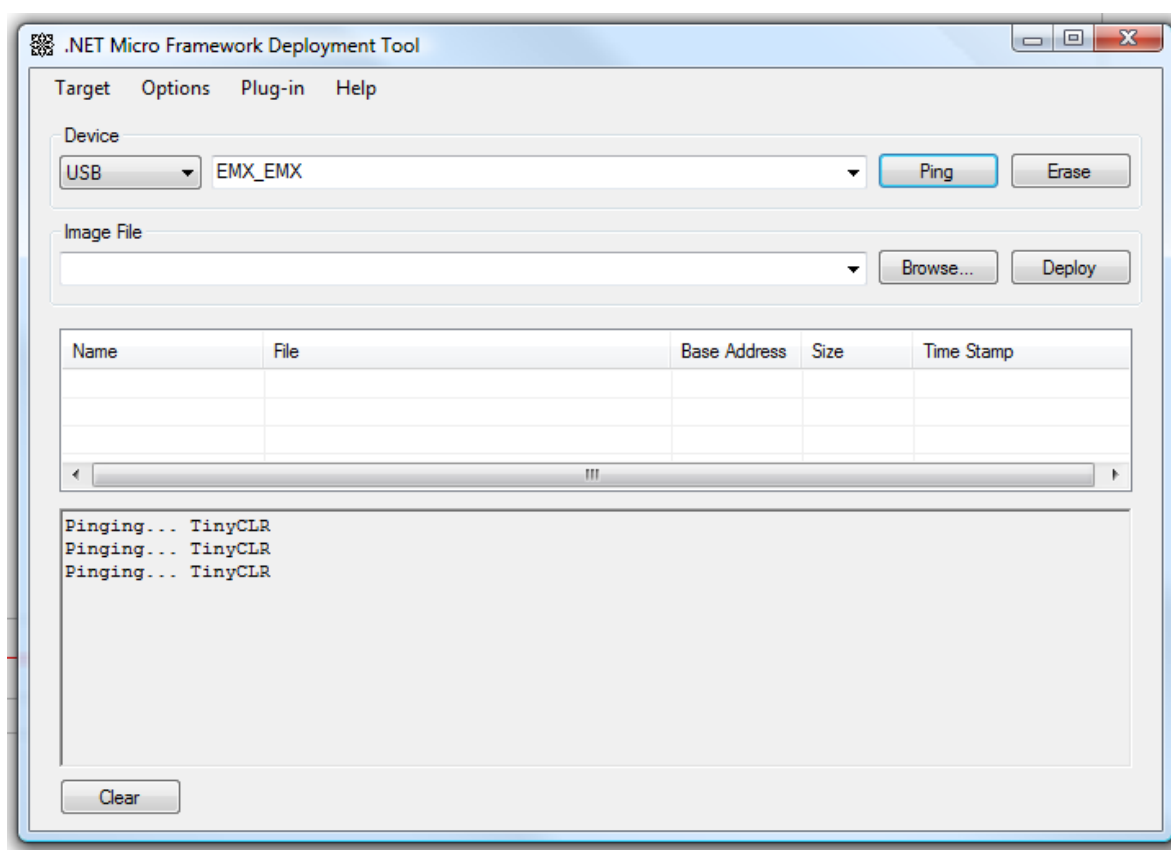


Important Note: The development system with the many peripherals, such as the TFT display, requires relatively high power. So make sure that you connect to a direct USB port at your PC or to a powered USB hub, or the board might not function correctly due to the insufficient supplied power. If you can not provide a USB port with good power or you are not sure. you can connect external power directly to the development system.

MFDeploy can Ping!

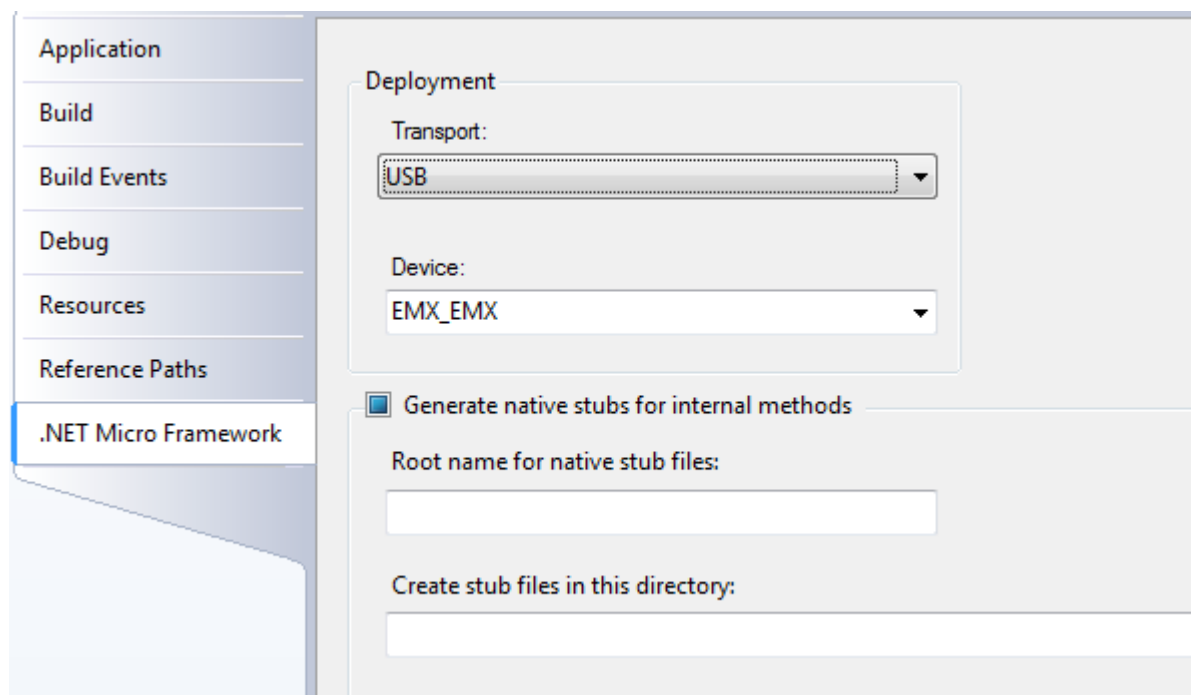
Before we use the hardware, let us make sure it is properly connected. The NETMF (.NET Micro Framework) SDK comes with a software from Microsoft called MFDeploy. There are many good uses for MFDeploy but for now we only need it to “ping” the device. Basically, “ping” means MFDeploy will say “Hi” to the device and then checks if the device will respond with “Hi”. This is to make sure the device is connected properly and communication has no issues.

Open MFDeploy and connect EMX using the included USB cable to your PC. If this is the first time you plug in EMX, Windows will ask for drivers. Supply the driver from the SDK folder and wait until windows is finished.



Deploying to Hardware

Now that we checked the hardware is working using MFDeploy, we need to go back to Visual C# express. From the project properties, select USB for transport and EMX for the device. Make sure your setup looks similar to the image below.



Allow a couple seconds to ensure the hardware has completed the boot up process then press F5, we will now send our simple application to EMX and it will run right inside the real hardware. Switching from emulator to real hardware is that simple!

Try the steps we tried with the emulator, like setting breakpoints and using F11 to step in the code. Note that "Debug.Print" will still forward the debug messages from the hardware to the output window on Visual C# express.

You have to keep in mind that only one software can talk to EMX interface at the same time. In another word, you can not Ping the hardware through MFDeploy if it is already connected through Visual Studio.

Connect MFDeploy to show debugging messages

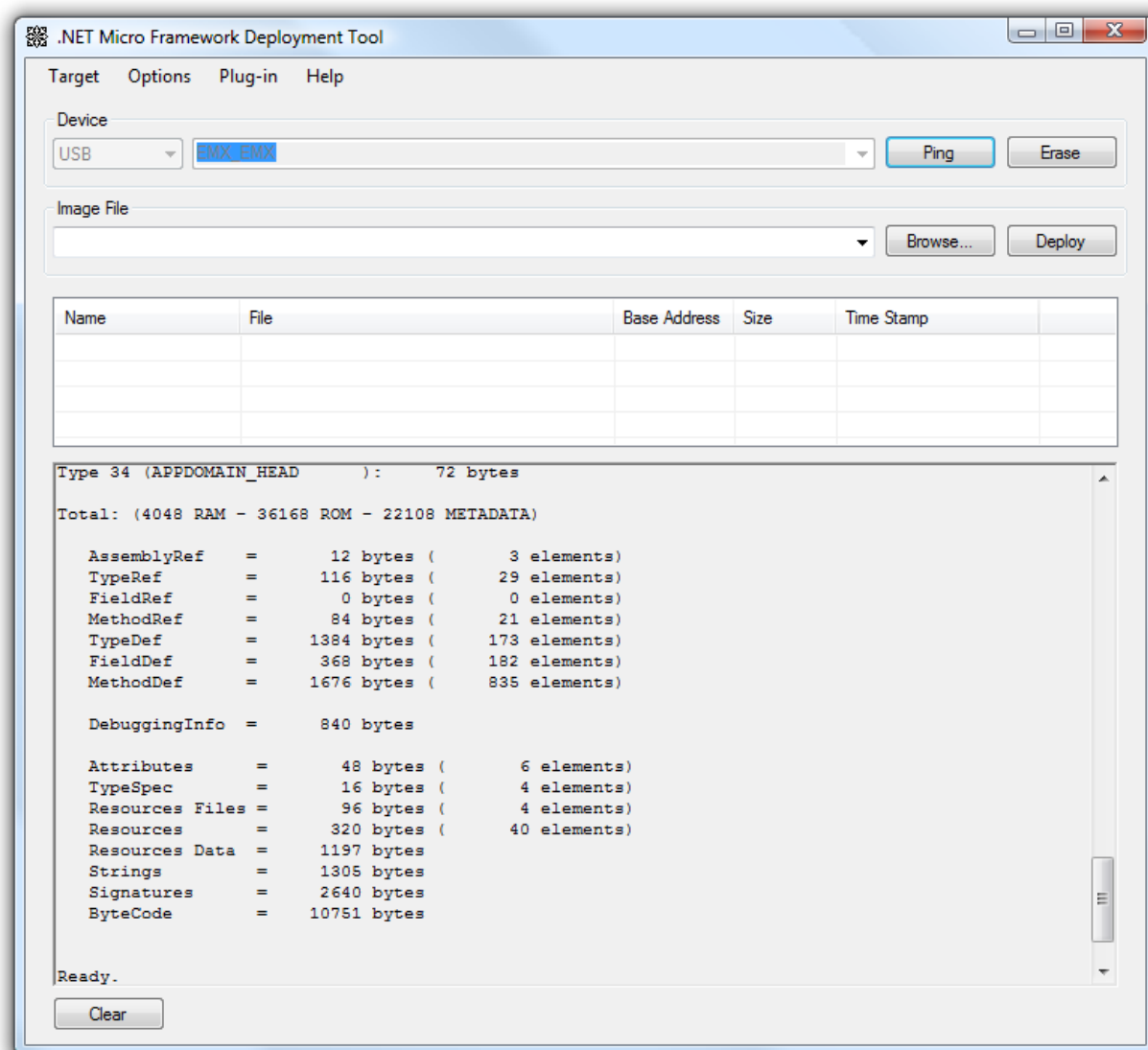
After Deploying the application through Visual Studio. The application still works on EMX even if is Visual Studio was disconnected and it will run every time you reset or recycle power on EMX.

User can still view the boot up and debugging messages by connecting MFDeploy to

USB. Simply after you connect USB cable and you can ping EMX in MFDeploy, click Target->Connect.

After this you can even reset the board and see all the boot up sequence messages and any debugging messages including the strings printed out in the code using `Debug.Print()`. In our previous example, "Amazing!". When done, you can disconnect using Target->Disconnect.

Important Note: If you did not see the debugging messages, press the Ping button and the debug messages will be displayed after.



Checking EMX firmware and TinyBooter Version

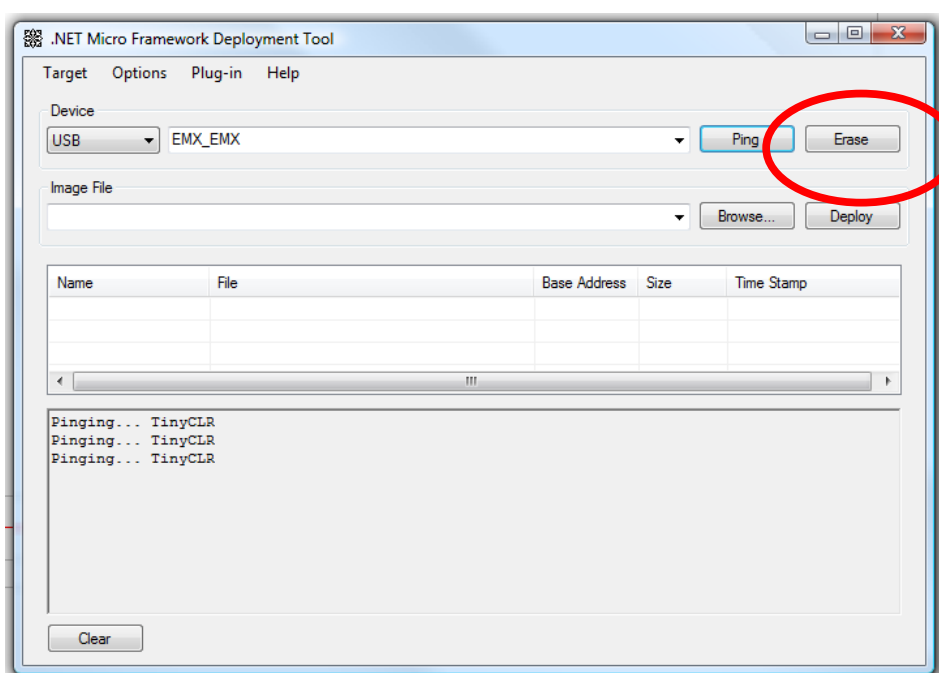
At this point and before proceeding, it is important to check if you need to update your EMX firmware. When you download a new SDK, it comes with new firmware that needs to be sent to EMX. Please see system update section at the end.

2.4. Deleting The Deployed Project

User can always over-deploy the current application (project) on EMX by deploying the newer application from Visual C#. but in some cases user might need only to delete the current application. this can be accomplished using MFDeploy tool explained before:

Open MFDeploy, choose USB and EMX from device list and ensure you get positive response when pinging

Then click the erase button:



Emergency user application code deletion

In some cases, user needs to delete the application because it is locking up the device or blocking the USB debugging interface. but Visual C# or MFDeploy tool does not help here and can not communicate with the device.

In this case you should access The Tinybooter mode (by pressing and holding Up and Down buttons on system bootup, you will see "Tinybooter on the screen", release the buttons the Erase the application code using MFDeploy as explained earlier.

3. Assemblies With EMX Project

3.1. What are Assemblies?

Hardware/Software features are accessed through Managed code using C# Classes with their members such as methods, properties and variables. On EMX you may need assemblies from 2 sources:

Standard Assemblies

.NET Micro Framework SDK includes pre-compiled assemblies that contain the managed code of various classes for hardware features such as Debug Class that you've just used in the example to print the string "Amazing!" using the Print Method. Users do not see the source code of these classes but can use these classes by simply adding the relevant assemblies to the Visual C# project.

.NET Micro Framework library examples:

- Microsoft.SPOT.Hardware
- Microsoft.SPOT.Native
- System.IO

Classes are described in .NET Micro Framework SDK documentation which also includes information about the required assemblies.

GHI NETMF assemblies

EMX hardware adds more functionality on top of the .NET Micro Framework. For example, hardware peripherals such as CAN, Analog converters, PWM,...etc are not directly supported in .NET Micro Framework. However, they can be easily used with EMX.

The assembly provides access to hardware peripherals, storage solutions, USB connectivity, and many other features. The assembly files and their documentation are included with GHI NETMF SDK. These assembly files must be added to your Visual Studio project in order to be able to use the extra features.

EMX uses the same assemblies available for ChipworkX and USBizi. This also means, you can switch from EMX to USBizi to ChipworkX without the need to change any line of code!

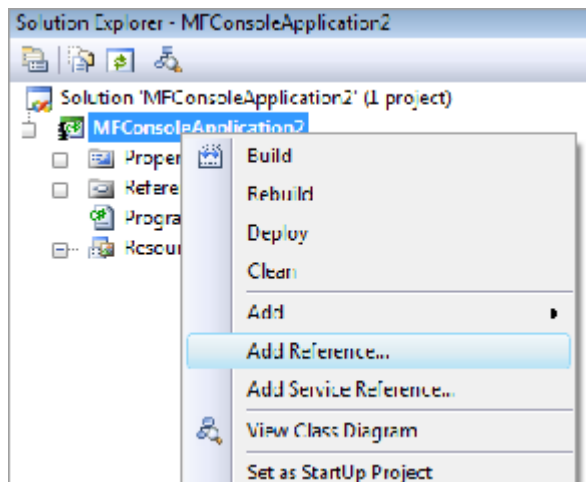
GHI NETMF library examples:

- GHIElectronics.NETMF.Hardware
- GHIElectronics.NETMF.System
- GHIElectronics.NETMF.Net

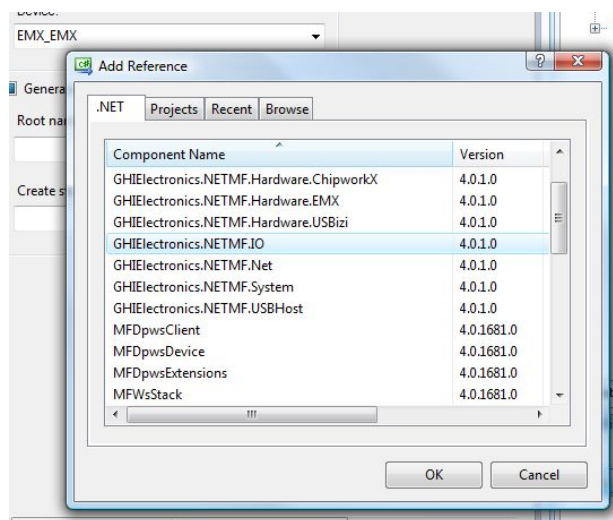
Adding Assemblies to Visual C# Project

Adding a library is pretty simple.

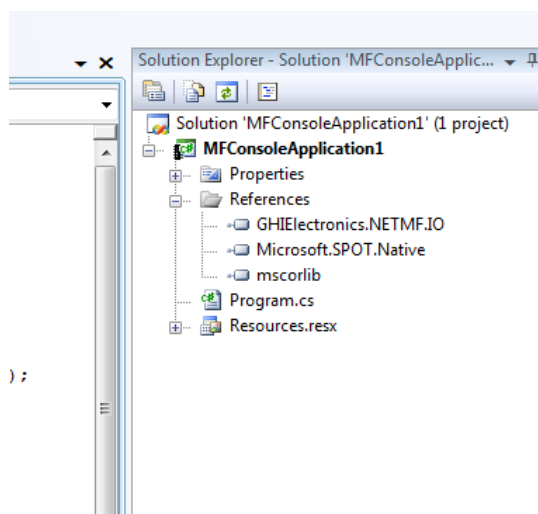
1. Go to the Menu and select “project --> Add Reference...”



2. Choose the library that you need then click OK.



3. We can now see the new Assembly reference with the References in the solution explorer.



4. TinyBooter And Firmware Update

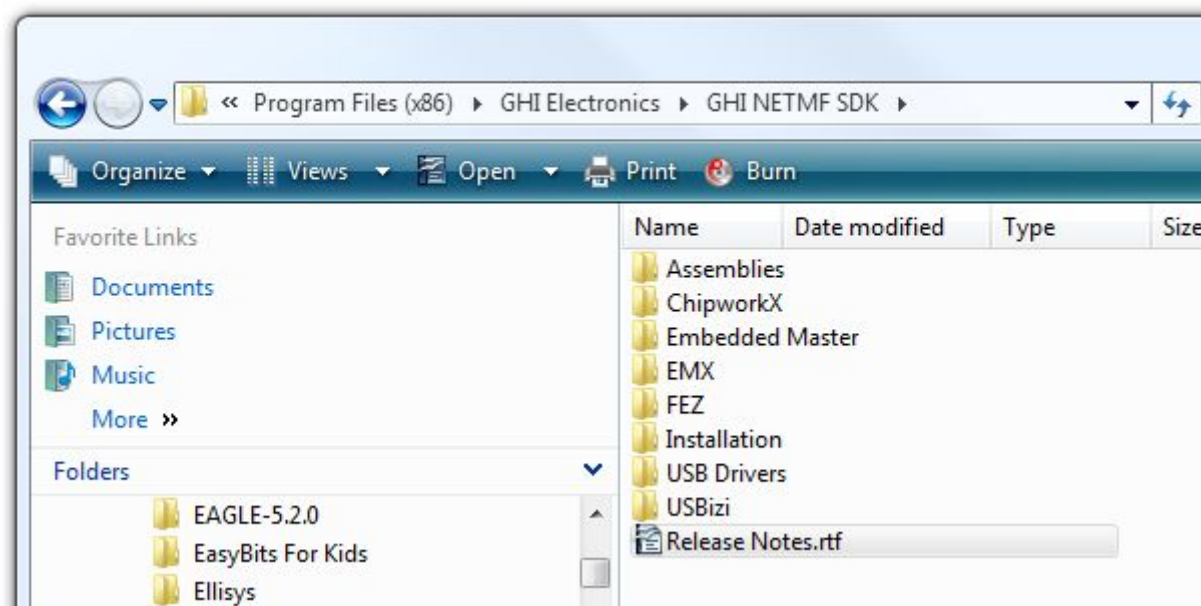
EMX includes three pieces of embedded software, GHI boot loader, TinyBooter and EMX firmware.

Important note: Always make sure the firmware loaded on the device matches the firmware version associated with the GHI NETMF SDK EMX firmware. Ideally, whenever you install a new SDK on your PC, you will also update the firmware on your device and update the assemblies you have added to your application. the next sections explains how to check the firmware version number and how to updating the firmware is explained here.

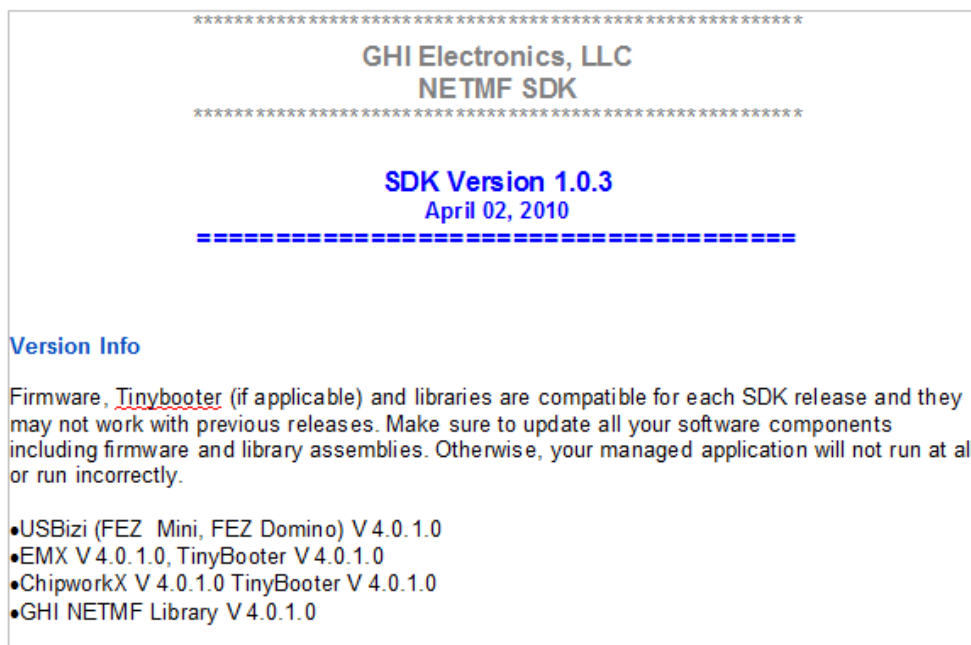
4.1. Checking the Firmware Version

Before updating the firmware, you can check the version number and make sure it needs to be updated or if you have the latest installed. Make sure you have the latest SDK and firmware installed.

Then go to www.ghielectronics.com (EMX downloads), the current GHI NETMF SDK version is shown in the download link. You can download and install the SDK. When done, the version release notes and changes are shown automatically. These are also available in the SDK installation folder.



After you have the SDK installed, you can see the Release Notes file, for example EMX firmware version number is 4.0.1.0 and Tinybooter version number is 4.0.1.0 in this SDK.



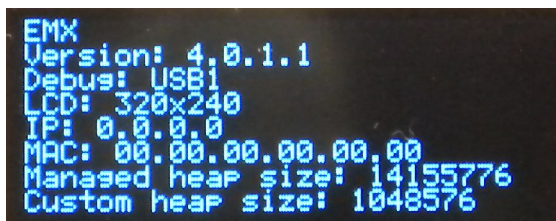
Version Info

Firmware, Tinybooter (if applicable) and libraries are compatible for each SDK release and they may not work with previous releases. Make sure to update all your software components including firmware and library assemblies. Otherwise, your managed application will not run at all or run incorrectly.

- USBizi (FEZ Mini, FEZ Domino) V 4.0.1.0
- EMX V 4.0.1.0, TinyBooter V 4.0.1.0
- ChipworkX V 4.0.1.0 TinyBooter V 4.0.1.0
- GHI NETMF Library V 4.0.1.0

The easiest way to verify the version numbers on EMX Development System is to read them on EMX's display.

On system boot up EMX firmware version number is printed out on second line of the screen.



```
EMX
Version: 4.0.1.1
Debus: USB1
LCD: 320x240
IP: 0.0.0.0
MAC: 00.00.00.00.00.00
Managed heap size: 14155776
Custom heap size: 1048576
```

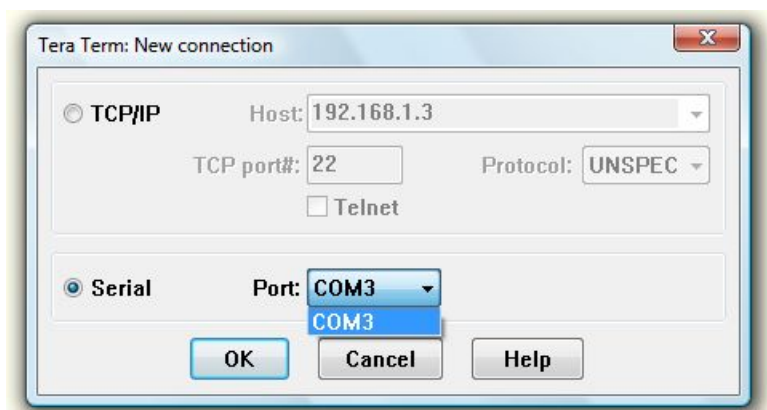
To read TinyBooter version number, all you have to do is to access TinyBooter (Press and hold Up and Down buttons on system boot up then release the buttons when you see "TinyBooter Mode" on EMX's display. The version number is right after that line.

4.2. TinyBooter Update through GHI Boot Loader

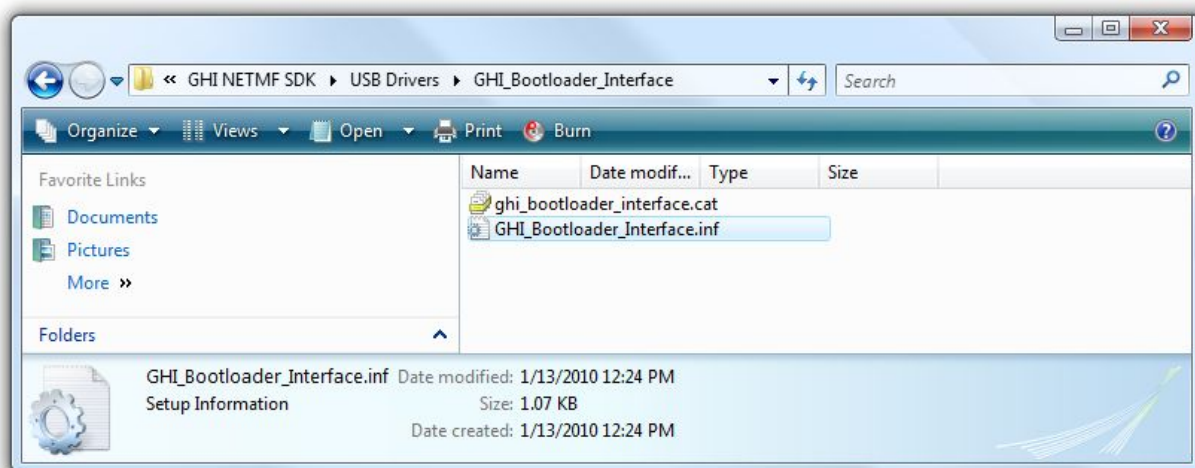
At power up, a GHI boot loader takes over the processor and validates TinyBooter stored in FLASH. If TinyBooter was found and was valid, execution is transferred to TinyBooter.

First, it is better to start fresh before loading the new firmware:

1. First, install the latest GHI NETMF SDK (which includes the latest EMX firmware and Tinybooter binary files).
1. Install TeraTerm software. We highly recommend you download and install from GHI website: [Download TeraTerm](#).
2. Disconnect the development system from power and from PC then open TeraTerm.
3. Select serial and click the arrow to drop down the list of COM ports. Note how many COM ports you have and write them down. On my system, I only have COM3.



4. Close TeraTerm for now.
5. Connect back the development system.
6. Press and Hold UP, Select and Down buttons the press and release Reset button.
7. Release the three buttons.
8. Windows will ask for a new driver. Driver file is GHI_BootLoader_Interface.inf



9. After windows is done installing the new drivers, open TeraTerm and observe what COM ports are now available. You should have a new COM port. This new COM port (serial port) is actually a USB connection between windows and GHI Boot loader. Windows applications do not know that this is a USB connection and treat it just like if it was a serial port.
10. Select the new COM port and click "OK". Now press the "b" button on the keyboard. You should see back BL for every time you press "b". Now try to press "V" (upper case) to get back the loader version number. **Note**, your version number might be different! Also, this version is **NOT** related to GHI NETMF SDK or firmware.

```
BL
BL
BL
BL
1.01
BL
```

11. You are now successfully accessing the bootloader.
12. Erase the flash memory using **E** command then press **Y** to confirm (this will take several seconds).

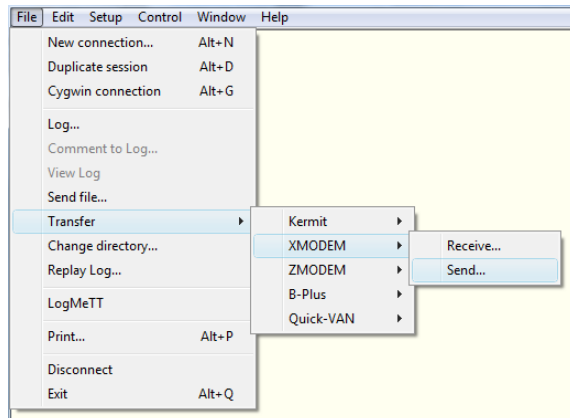
13. Loading new firmware is simple but it requires a terminal that supports XMODEM file transfer. XMODEM has many versions, GHI boot loader requires 1K transfers with 16Bit CRC error checking. Keep on using TeraTerm software.

Transfer is initiated using the **X** command. After the **X** command is entered, the GHI boot loader will start sending back the “C” character continuously. This “C” is an indicator that tells XMODEM a device is waiting for data. Once you see the “C” character appearing on the terminal window, you can select XMODEM transfer and point the software to the firmware file "TinyBooter.GHI."

Entering **X** command...

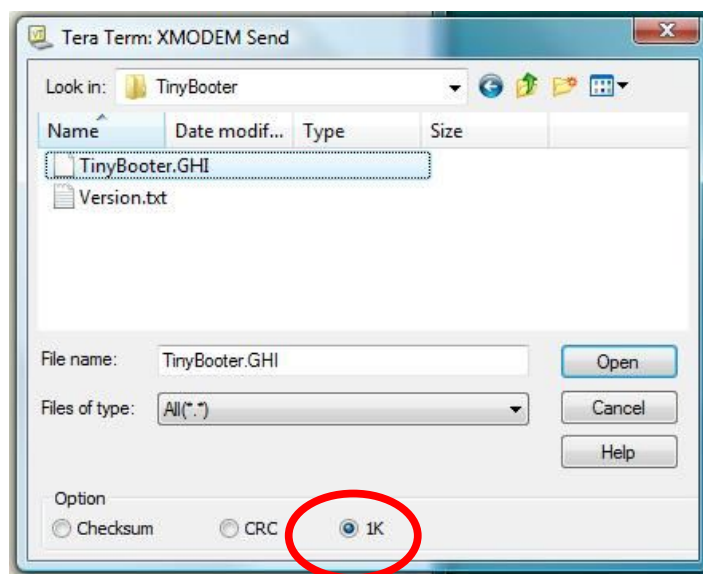
```
BL
BL
BL
BL
1.01
BL
Start File Transfer
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

In the menu, select **File > Transfer > XMODEM > Send...**



Next, select the **TinyBooter.GHI** file from...

GHI Electronics\GHI NETMF SDK\EMX\Firmware\TinyBooter



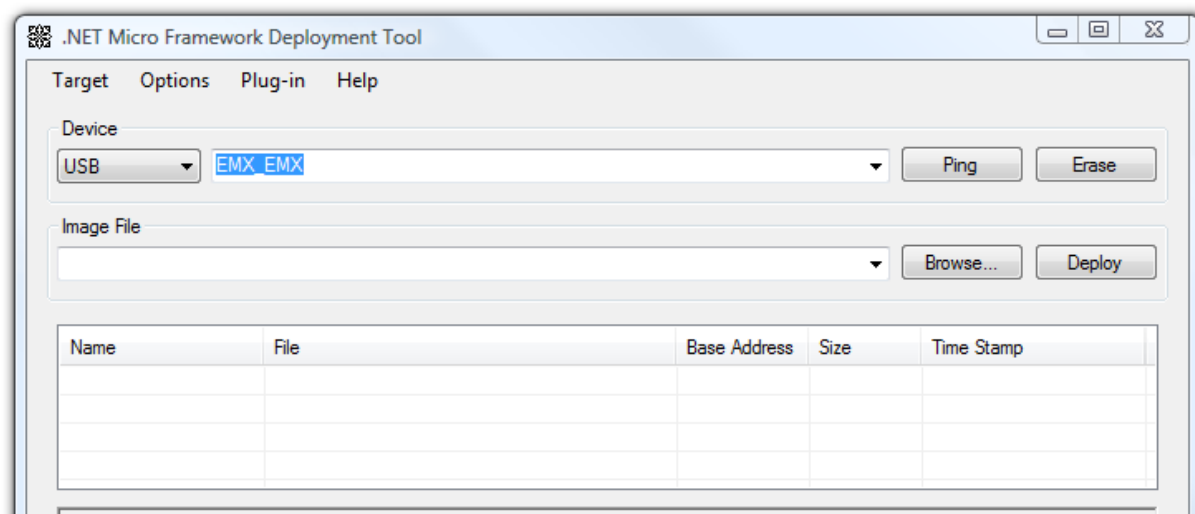
Updating the firmware takes a few seconds to load. Once loading has finished and the file is valid, the new firmware is executed automatically and you will not see "BL" again. If you did see the "BL" again after this process, it means that the update process failed. You might need to double check that the downloaded file is not corrupted.

At this point you have a clean Flash memory with only Tinybooter waiting to get the new EMX firmware files to update the system.

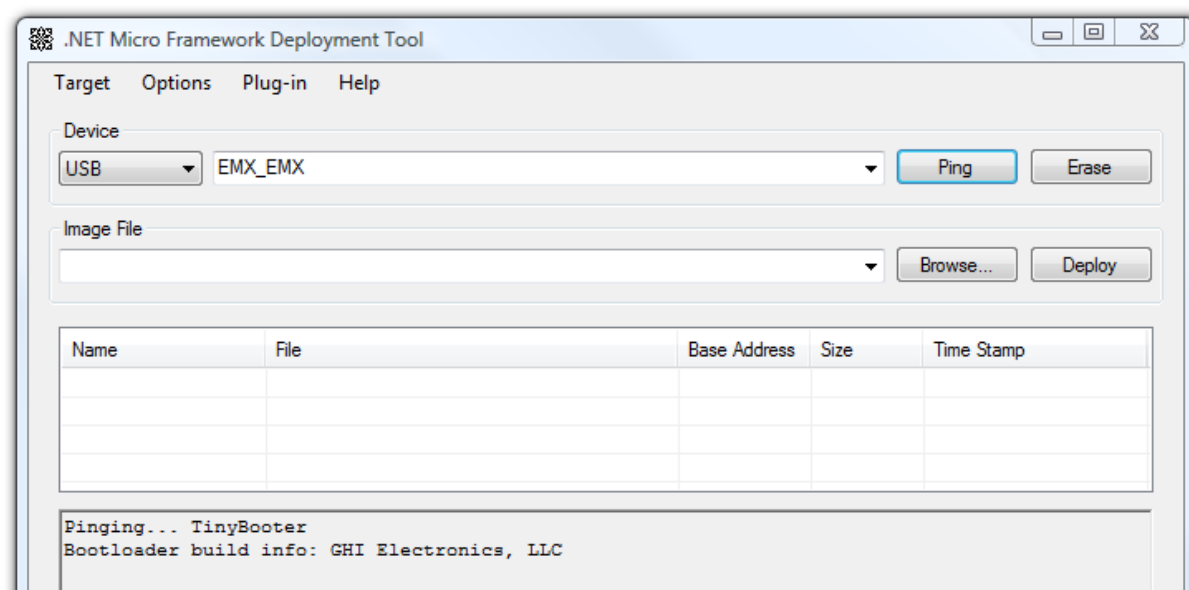
4.3. EMX firmware update through TinyBooter

In the following steps we will update EMX firmware with Tinybooter and MFDeploy help:

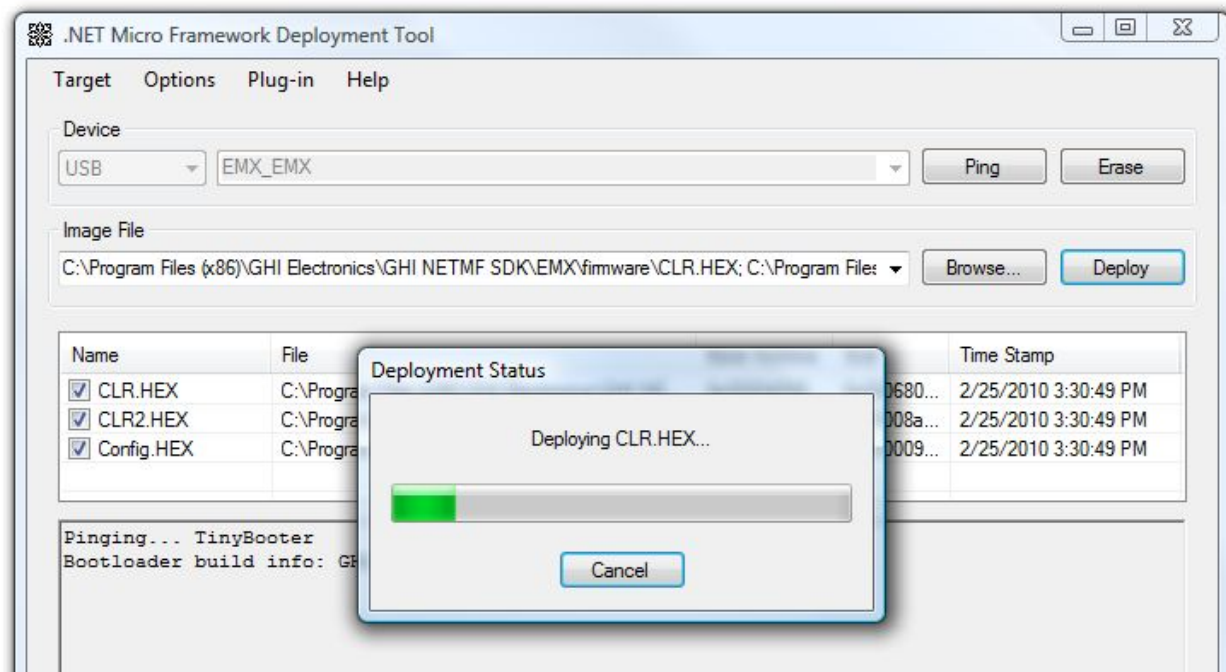
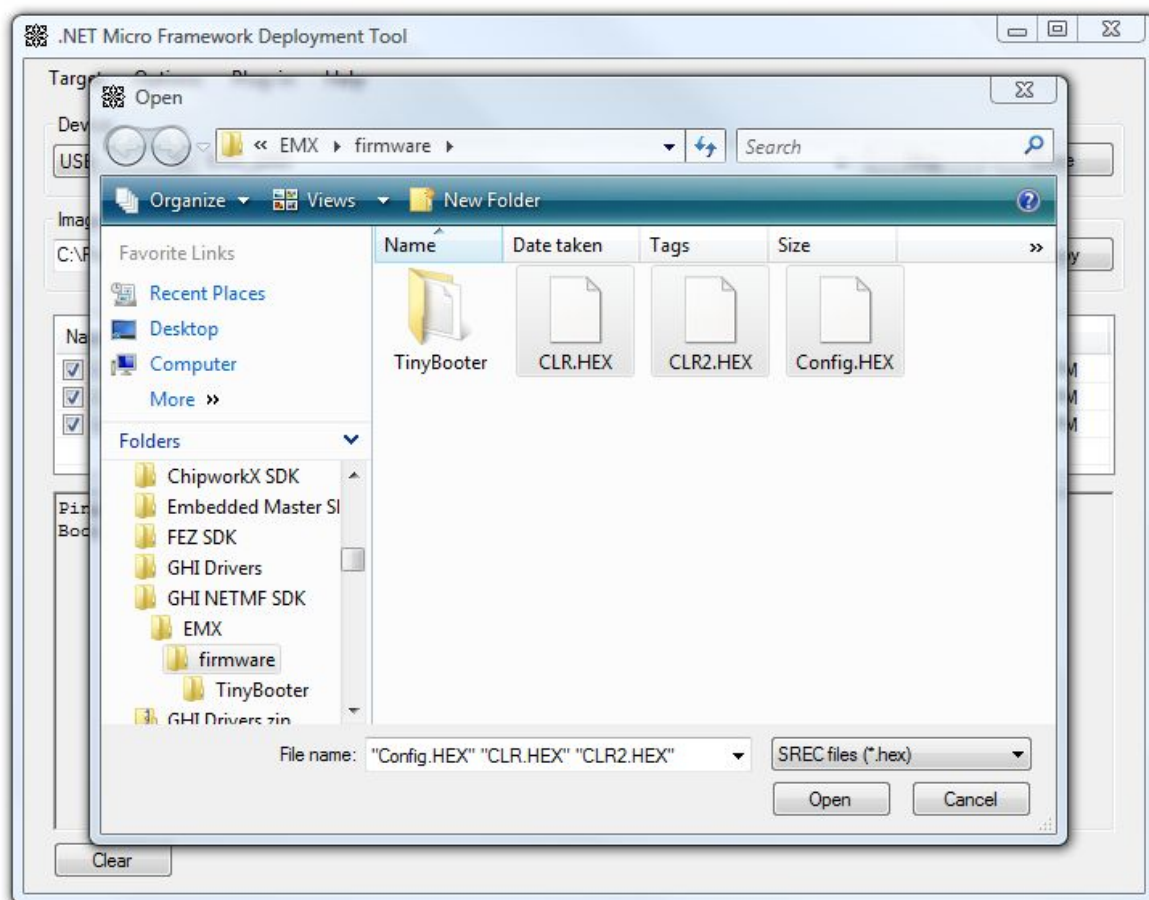
2. First, install the latest GHI NETMF SDK (which includes the latest EMX firmware and TinyBooter binary files). You probably already did when you updated the TinyBooter in the previous section.
3. Insure there is no need to update the TinyBooter or you already did.
4. Press and hold the Up and Down buttons then press and release reset to access TinyBooter mode. You will see on the EMX's TFT display.
5. Run MFDeploy and select USB from the Device list, you should see EMX_EMX in the dropdown.



6. Check the communication between MFDeploy and TinyBooter by pinging the device. Press Ping and you should see this message:



7. Now we can lead MFDeploy to the new EMX firmware files. Click Browse and direct MFDeploy to the firmware HEX files. These can be found under EMX\firmware folder in the SDK. The other files with “sig” extension must exist in the same folder as the HEX files. Select **ALL** of the HEX files at once and start deploying the firmware by pressing Deploy.



8. Loading the files takes about a minute. On completion the firmware will execute. Double check the version number to make sure the correct firmware is loaded.
9. Loading new firmware will not erase the deployed managed application. If you need to erase the managed application click Erase.

5. What Is Next?

There are many documents and resources that helps you out with EMX and .NET Micro Framework:

- EMX User Manual
- EMX downloads page on www.ghielectronics.com
- GHI blog is always a good place to visit
<http://tinyclr.blogspot.com/>
- The Micro Framework Project website is an excellent resource
<http://www.microframeworkprojects.com/>
- A good and free eBook to continue learning about C# is available at
<http://www.programmersheaven.com/2/CSharpBook>
- Jens Kuhner excellent book on .NET Micro Framework
<http://www.apress.com/book/view/9781430223870>
- USB complete is an excellent book on USB
<http://www.lvr.com/usbc.htm>
- .NET Micro Framework's main page on Microsoft's website
<http://www.microsoft.com/netmf>
- .NET Micro Framework Community
<http://www.netmf.com/>

Legal Notice

Licensing

EMX Module is fully licensed for commercial use. The Module price covers the commercial use of EMX Module with .NET Micro Framework.

Disclaimer

IN NO EVENT SHALL GHI ELECTRONICS, LLC. OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS PRODUCT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. SPECIFICATIONS ARE SUBJECT TO CHANGE WITHOUT ANY NOTICE. GHI ELECTRONICS, LLC LINE OF PRODUCTS ARE NOT DESIGNED FOR LIFE SUPPORT APPLICATIONS.

EMX is a Trademark of GHI Electronics, LLC

.NET Micro Framework, Visual Studio, MFDeploy, Windows Vista, Windows SideShow are registered or unregistered trademarks of Microsoft Corporation.

Other Trademarks and Registered Trademarks are Owned by their Respective Companies.